Code Signing—Get Used to it!

MacTech Magazine
November • 2008

MACTECH®

The Journal of Macintosh Technology

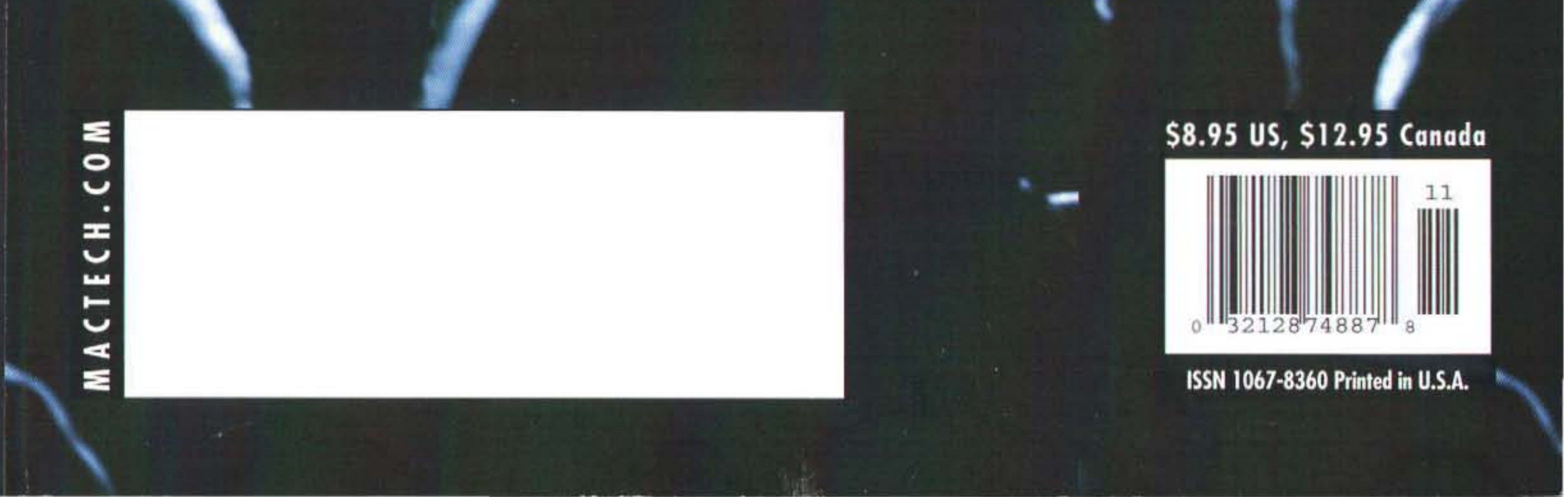
MacTech's Geek Guide to Buying Projectors

Real World Reviews:

Da-Lite Deluxe Insta-Theater • NEC NP3151W Projector

MCX: No Excuses, Now!

Integrating OS X With OpenLDAP/Samba



MACTECH.COM

$8.95 US, $12.95 Canada

ISSN 1067-8360 Printed in U.S.A.

ETYMOTIC
hf2
headset + earphones
iLounge.com
BEST of SHOW
2008

MacMall

Your #1 Apple Superstore!

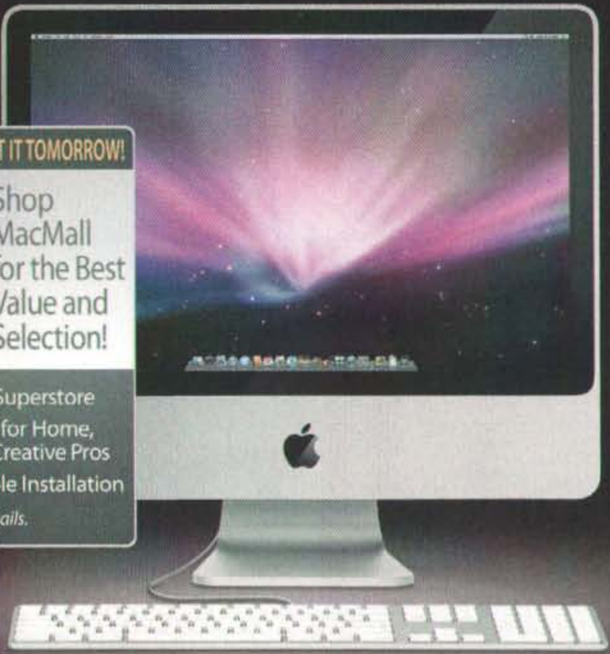
Shop MacMall for the Best

ORDER TODAY—GET IT TOMORROW!

Shop MacMall for the Best Value and Selection!

- Your #1 Apple Superstore
- Apple Experts for Home, Business and Creative Pros
- Windows/Apple Installation

Click for all the details.



MacMall Exclusive Deals!

Get up to $250 cash back!*

FREE Parallels Desktop 3.0!*

FREE Epson All-in-One Inkjet Printer!*

See below for details.

iMac®

- FASTER! Up to 3.06GHz Intel Core 2 Duo processor
- NEW! Up to 2GB 800MHz DDR2 SDRAM
- Up to 500GB hard drive ■20" or 24" glossy widescreen LCD
- 8X SuperDrive® with dual layer

~~$1194~~ – $50 mail-in rebate

iMac starting at **$1144!** #7497733

Finance for $36/mo.

Call or visit our Web site for the best selection of add-ons for your Mac!

AXIOM

2GB DDR2 SDRAM for Select iMac, MacBook and MacBook Pro

only $54.99! #7119342

Call for a perfect match to your Mac!

EPSON

Stylus CX4450 Color All-in-One Printer

~~$79.99~~ FREE! #7525176

After $50 mfr. and $29.99 MacMall mail-in rebate with Apple CPU purchase. Price before rebates is $79.99. Ends 10/31/08. See below.†

Apple AirPort Express™ Base Station with 802.11b/g and AirTunes™

only $59.99! #448199

Kingston

8GB USB 2.0 Flash Drive

only $22.99! #7383178

WD Western Digital®

250GB My Passport Studio HD

only $124.98! #7490963

Apple 1TB Time Capsule

only $494! #7373107

Brenthaven®

Trek Sleeve Laptop Case for MacBook Air & 13.3" MacBook

only $22.99! #7497466

LaCie

1TB Big Disk Extreme Hard Drive

only $219.94! #7274662

Logitech

AudioHub Notebook Speakers and Hub

only $45! #7513931

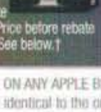
Kensington

MicroSaver Notebook Lock

only $34.99! #7052823

VMware Fusion for Mac

#7288411



~~$69.95~~ $19.95!

After $50 mfr. mail-in rebate with Apple CPU purchase. Price before rebate is $69.95. Ends 12/31/08. See below.†

MacSpeech Dictate

#7372255

only $164.99!

Jawbone Bluetooth Headset

#7296308

Aliph™

only $79.99!

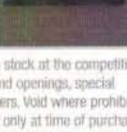
Apple® Component AV Cable

#7298864

only $49!

Mac OS® X v10.5.4 Leopard™

#7531897



only $109.98!

*WE'LL BEAT ANY PRICE ON ANY APPLE BRAND PRODUCTS or competitive promotion on any Apple computer. We may require proof of dealer's pricing. Competitor must be an Apple Authorized Reseller. Product must be in stock at the competition. Advertised item must be identical to the one offered by MacMall and must be a new item in a factory sealed box. We reserve the right to limit quantity ordered. This offer does not apply to special, bonus or free offers, grand openings, special purchases or special buys, manufacturer's rebates, closeouts, clearances or to dealers' one-of-a-kind or other limited quantity offers. Nor does it apply to advertising errors made by any authorized dealer. Not open to dealers. Void where prohibited. Sorry, offer not valid in conjunction with other rebate offers. On multiple-item quotes: "We'll Beat Any Price" applies to the price of the entire order, not individual line items. Offer applies to pre-taxed prices only. Offer valid only at time of purchase, not after the fact. ■ UP TO $250 CASH BACK OFFER-Get up to $250 Cash Back via MacMall mail-in rebate with purchase of select computer models. Ends 10/7/08. ■ FREE PRINTER OFFER-Get an Epson Stylus CX4450 All-in-One Inkjet

NEW iPods models announced by Apple!
Plus... NEW Adobe Creative Suite 4 available now! *Call today to order!*

Value & Selection!

Authorized Reseller

Mac® Pro

- Up to 8-core performance with up to 2 Quad-Core Intel Xeon 5400 processors at up to 3.2GHz
- ATI Radeon HD 2600 XT graphics with 256MB GDDR3
- Four internal drive bays support up to 4TB of storage

~~$2294~~ - $125 mail-in rebate
starting at **$2169!*** #7371990

MacBook™

- Up to 2.4GHz Intel Core 2 Duo processor with 45nm technology
- Up to 250GB hard disk drive
- Up to 2GB DDR2 SDRAM
- 13.3" glossy widescreen display
- 802.11n wireless ■ iSight camera

~~$1094~~ - $75 mail-in rebate
starting at **$1019!*** #7404921

MacBook™ Air

- Up to 1.8GHz Intel Core 2 Duo processor ■ Only 0.76" thin
- Weighs just 3lbs.!
- 80GB HD or 64GB solid-state drive
- 802.11n wireless ■ iSight camera
- 13.3" glossy widescreen backlit LCD

~~$1794~~ - $50 mail-in rebate
starting at **$1744!*** #7373085

MacBook™ Pro

- Up to 2.6GHz Intel Core 2 Duo processor with 45nm technology
- Up to 6MB processor cache
- 17" model with arsenic-free glass and optional mercury-free LED backlit display

~~$1994~~ - $150 mail-in rebate
starting at **$1844!*** #7405626

Mac® mini

- Intel Core 2 Duo processor at speeds up to 2GHz
- 1GB SDRAM ■ Up to 120GB HD
- SuperDrive™ with dual-layer support or Combo drive
- 802.11g wireless & Gigabit Ethernet

~~$594~~ - $25 mail-in rebate
starting at **$569!*** #7288184

1-800-MACMALL (1-800-622-6255) **www.macmall.com** Source Code: MACTECH

You can't beat these exclusive prices!

15.4" Glossy Display,
2.2GHz Intel Core 2 Duo processor

- 2GB SDRAM ■ 120GB hard drive
- 8X SuperDrive (DVD+R DL/DVD±RW/CD-RW)

original price $1,994
~~$1599.99~~ - $150 mail-in rebate
$1449.99!* #7256720

15.4" Glossy Display,
2.4GHz Intel Core 2 Duo processor

- 2GB SDRAM ■ 160GB hard drive
- 8X SuperDrive (DVD+R DL/DVD±RW/CD-RW)

original price $2,494
~~$1798.99~~ - $150 mail-in rebate
$1648.99!* #7256721

Model	Description	was	now	MacMall rebate	Price	Finance
#7405161 *glossy!*	15.4" 2.4GHz Intel Core 2 Duo, 4GB RAM, 200GB HD	~~$2994~~	~~$2399~~	$200	$2199!*	$72/mo
#7349141	15.4" 2.6GHz Intel Core 2 Duo, 2GB RAM, 160GB HD		~~$2399~~	$250	$2149!*	$72/mo
#7349144	17" 2.6GHz Intel Core 2 Duo, 2GB RAM, 160GB HD		$2649	$250	$2399!*	$80/mo

intel Core 2 Duo

Parallels Desktop 3.0 for Mac
#7257816
~~$79.99~~
FREE!
After rebate with Apple computer purchase. See below.†

Office 2008 for Mac Home and Student Edition
#7352258
Microsoft
only **$124.98!**

Photoshop Lightroom 2
#7507304
Adobe
New!
only **$278.99!**

Norton Antivirus 11.0 for Mac
#7352086
symantec.
only **$47.99!**

Apple MobileMe
#7516882
New!
only **$79.98!**
Family Pack only $119.98! #7516883

Printer (#7525176) FREE after $50 mfr. and $29.99 MacMall mail-in rebate with purchase of any Apple computer. Price before rebates is $79.99. Ends 10/31/08. ■ FREE PARALLELS DESKTOP 3.0 OFFER-Get Parallels Desktop 3.0 for Mac (#7257816) FREE after $20 mfr. and $60 MacMall mail-in rebates with purchase of any new Intel-based Apple CPU. Price before rebates is $80. Ends10/7/08. ■ ALL OFFERS VALID ONLY WHILE SUPPLIES LAST. †For rebate terms and conditions, please visit our Web site and enter the applicable part number. Download rebate coupons at www.macmall.com/rebates.

TABLE OF CONTENTS

From the Editor

The Apple universe may be a bit chaotic as of late—stock price swaying, app store antics, etc.—but the underlying technology is sound. Of course, it's the techs that implement systems that keep all of this the rock that doesn't sway in the face of chaos. This month, MacTech brings you some very rock-solid foundational articles.

This month's Mac in the Shell column focuses on learning Python. Python has been around for over 15 years, but is only recently getting heavy use on OS X and OS X Server. I consider scripting an absolute foundation of being a System Administrator. We've covered bash pretty extensively, and, while capable, bash has its limits. Start learning Python on the Mac for general or System Administration use.

Noah Gift brings us a multi-part article on integrating OS X with other directory services. This first article lays the groundwork for the following articles, and makes sure you have what you need to begin testing.

Dave Dribin and the Road to Code covers one of the hallmarks of the Mac since the very early days of the GUI: Undo support. This is an important next step for developers to bring a polished experience to users of their software.

New author Scott Corley presents an article that should be of interest to developers and System Administrators alike: all about code signing. Code signing has become more important to understand under Leopard, and for applications that run on the iPhone. This article explains what it is and how to dig in.

Our Geek Guide this month talks about projectors and setting up a projection environment. With all of the technology involved getting more affordable, it's a great time to understand all of the variables that that go into a system before you make a purchase.

I may have mentioned this before, but I'll take the risk of repeating this tidbit: System Administrators: you should be reading every word that Greg Neagle has to say. This month, Greg talks about the changes in options for managed preferences that came to us in Leopard. This is certainly another foundation that every OS X administrator should understand.

On the slightly more cutting edge, the third C4 conference, C4[2], recently took place in Chicago. Regular contributing author Marcus Zarra was one of the many attendees in attendance. If you don't know what C4 is, or do and wish you had been there, check out Marcus' report on the conference.

The MacTech Spotlight this month shines on Blair Yakimovich from Transgaming, Inc. Transgaming has developed a technology that they've dubbed "Cider." Essentially, Cider is a Win32-API emulator that allows developers to release games written for Windows for Intel-based Macs with little to no effort or changes in code. Consider it Rosetta for games. The technology is pretty amazing, and so are the people working on it. We talk to one of them in this month's MacTech Spotlight.

That's not all we have this month, so, crack open the rest of the issue, and enjoy! Also, don't forget to make your plans for Macworld. We hope to see everyone in San Francisco at the Mactech booth during the show!

Ed Marczak,
Executive Editor

Communicate With Us

Department E-Mails

Orders, Circulation, & Customer Service
cust_service@mactech.com

Press Releases
press_releases@mactech.com

Ad Sales
adsales@mactech.com

Editorial
editorial@mactech.com
(Authors only, no pr)

Accounting
accounting@mactech.com

Marketing
marketing@mactech.com

General
info@mactech.com

Web Site
http://www.mactech.com

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail?**

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact MacTech Magazine Customer Service at 877-MACTECH

We love to hear from you! Please feel free to contact us with any suggestions or questions at any time.

Write to letters@mactech.com or editorial@mactech.com as appropriate.

MACTECH®

The Journal of Macintosh Technology

A publication of **XPLAIN**CORPORATION

The Magazine Staff

Publisher & Editor-in-Chief: Neil Ticktin
Executive Editor: Edward R. Marczak
Business Editor: Andrea Sniderman
Ad Director: Ken Spencer
Production: David Allen
Staff Writer: Jeffrey A. Rochin
Staff Writer: Kelly Honig

Xplain Corporation Senior Staff

Chief Executive Officer: Neil Ticktin
President: Andrea J. Sniderman
Accounting: Marcie Moriarty
Customer Relations: Susan Pomrantz

Columnists

Reviews/KoolTools: by Michael R. Harvey
The Road to Code: by Dave Dribin
KoolTools/Geek Guides: by Dennis Sellers
Mac In The Shell: by Ed Marczak
Andy Inhatko
MacEnterprise: by Philip Rinehart and Greg Neagle

Regular Contributors

José R.C. Cruz, Doug Hanley, Mary Norbury, Norman Palardy, Andy Sylvester, Rich Warren, Ryan Wilcox, Marcus S. Zarra

Canada Post: Publications Mail Agreement #41513541
Canada Returns to be sent to: Bleuchip International, P.O. Box 25542, London, ON N6C 6B2

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are $47.00 per year. Canadian subscriptions are $59.00 per year. All other international subscriptions are $97.00 per year. Domestic source code disk subscriptions are $77 per year. All international disk subscriptions are $97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

C4 (2) Conference

a report on this year's event

by Marcus S. Zarra

On September 5, 2008, the third C4 kicked off; titled C4[2]. C4 is a conference hosted by a Macintosh independent developer aimed at the world of Macintosh development. When you attend this conference you know that you are surrounded by the best and brightest of the Macintosh developer community. While the first two of the annual C4 conference were well received, they also set the bar very high for this year. Fortunately our host, Wolf Rentzsch, was up to the task.

Friday evening started out with a talk from Craig Hockenberry and was followed by the creator of SQLite, Dr. Richard Hipp. Craig's talk about his process for developing Twitterific was quite insightful and Dr. Hipp's discussion of SQLite was spectacular. I suspect even more applications will be utilizing SQLite in the near future.

Saturday promised to be an extremely long day with 7 talks lined up and finishing off with a panel of experts. Personally, I found Rich Siegel's talk quite interesting. A Discussion of Macintosh development from the perspective of someone who has been doing it for over 15 years was very insightful. In the afternoon, the session presented by a group of "Security Researchers" was both stunning and terrifying. They pressed so much information into their block of time that I will need to watch the videos at half speed just to consume what they were trying to explain to us. Brent Simmons wrapped up the day's sessions with a talk on how and why he took NetNewsWire from a successful paid-for application to a free (as in beer) application.

At the end of Saturday night, Wil Shipley hosted a panel of experts to discuss a wide range of topics surrounding the current world of Macintosh development. While he had a list of his own questions he also was fielding questions from the audience via twitter. The answers to the questions were quite informative and the panel reminded us that there are always multiple solutions to any issue.

Sunday fortunately ran at a slower pace and only had 3 talks scheduled, although all three were not to be missed. Mike Lee's discussion on how to "pimp" your applications was quite note-worthy. Andy Finnell's talk on being an independent contractor reminded us that there are many ways to be an independent developer on the Macintosh platform. Troy Gaul discussed how Lightroom came about and gave a terrific insight into the inner workings of Adobe.

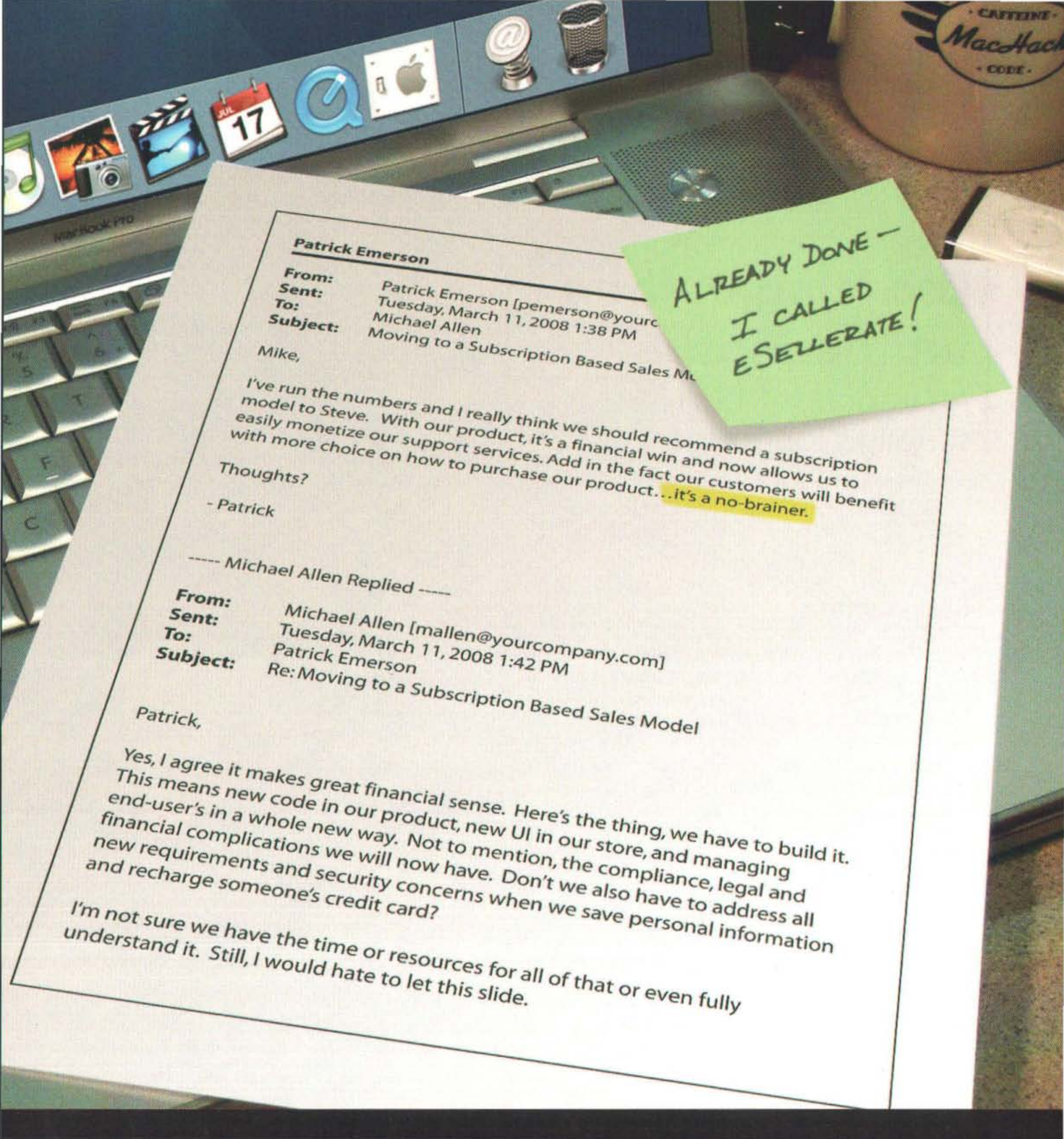
Sunday was wrapped up with the presentation of the Iron Coder entries. This year's theme was on Paranoia and the API was Core Location. Two of the entries centered around the crime in your local area and reminded all of us that Chicago is not the safest city to be in. One developer who lacked an iPhone decided to do a clean room implementation of Core Location on the desktop; a simply amazing accomplishment. In the end though, when the voting was done, Glen & Ken Aspeslagh of Ecamm Network ended up winning the first place prize with their combination of two iPhones, a Wii remote and their MacBook Pro.

Unfortunately not everything at C4[2] was perfect. The conference definitely suffered from its own success. This year, the conference was sold out in less than 40 hours and thanks to some negotiating by Wolf, it was expanded slightly to allow a fraction of the waiting list to be added. While at the conference you could definitely tell that we were at capacity. If you did not get to a session early enough you were left standing at the back of the room as there were simply not enough chairs to go around. This also caused some hotel issues and the Internet access was spotty. Hopefully next year the venue can be expanded some how.

Overall the conference was definitely worth going to. I even overheard some other attendees commenting that if they had to choose between WWDC and C4 that they would choose C4. High praise indeed!

About The Author

Marcus S. Zarra is the owner of Zarra Studios, based out of Colorado Springs, Colorado. He has been developing Cocoa software since 2003, Java software since 1996, and has been in the industry since 1985. Currently Marcus is producing software for OS X. In addition to writing software, he assists other developers by blogging about development and supplying code samples.



Patrick Emerson

From: Patrick Emerson [pemerson@yourc
Sent: Tuesday, March 11, 2008 1:38 PM
To: Michael Allen
Subject: Moving to a Subscription Based Sales M

Mike,

I've run the numbers and I really think we should recommend a subscription model to Steve. With our product, it's a financial win and now allows us to easily monetize our support services. Add in the fact our customers will benefit with more choice on how to purchase our product...it's a no-brainer.

Thoughts?

- Patrick

----- Michael Allen Replied -----

From: Michael Allen [mallen@yourcompany.com]
Sent: Tuesday, March 11, 2008 1:42 PM
To: Patrick Emerson
Subject: Re: Moving to a Subscription Based Sales Model

Patrick,

Yes, I agree it makes great financial sense. Here's the thing, we have to build it. This means new code in our product, new UI in our store, and managing end-user's in a whole new way. Not to mention, the compliance, legal and financial complications we will now have. Don't we also have to address all new requirements and security concerns when we save personal information and recharge someone's credit card?

I'm not sure we have the time or resources for all of that or even fully understand it. Still, I would hate to let this slide.

ALREADY DONE —
I CALLED
ESELLERATE!

eSellerate. Complete.

Begin defining your strategies at http://www.esellerate.net/mactech

eSellerate, the same team that brought you Installer Vise.

eSellerate is a registered trademark of MindVision, Inc. - a Digital River Company.

Mac Universal

MAC IN THE SHELL

by Edward Marczak

Learning Python on the Mac

An important, built-in scripting language

Introduction

Python is a scripting language that breaks the mold, in many respects. It is fully object-oriented, although you're not forced into the "fully" part. It uses indentation to denote blocks of code, and, it's named after *Monty* Python. It's an *ideal* scripting language for OS X. The next few Mac in the Shell columns will be dedicated to learning Python using OS X. If you're new to Python, or rusty on the basics, here's the place to start.

History

Although Python is enjoying a bit of a surge in popularity, it was created over 15 years ago, in 1990 by Guido van Rossum, and is quite mature. Again, think Monty Python, the comedy troupe, not the reptile variety. Python is designed to be relatively simple to learn. Also, thanks to being open source, there's a version of Python for just about any platform you can think of (and if there isn't already, *you* can get it running there, too!). You can leverage your Python skills wherever there's a Python interpreter, be it Windows, Linux, OS X or portable device.

OS X Tiger ships with Python version 2.3, and Leopard ships with version 2.5 (2.5.1, to be specific). If you need to use version 2.4 for compatibility with existing code, or for your company's standards, source and binary installers are available. See http://www.pythonmac.org/packages/ for a start, if needed.

Jumping In

Interestingly, much like the bash shell that this column has covered, the Python binary is both a run-time interpreter and an interactive shell. I'm going to present these examples by using bash in Terminal.app for now, and get into editors in future columns. So, let's jump in! Start by opening Terminal.app (found in Applications > Utilities). Type **python** and press return:

```
$ python
Python 2.5.1 (r251:54863, Apr 15 2008, 22:57:26)
[GCC 4.0.1 (Apple Inc. build 5465)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

The python interpreter prints some introductory information, and then leaves you at a prompt. The triple greater-than prompt lets you know that you're in Python's interactive shell; python is now "listening." Test this out (what you type is in **bold**):

```
>>> 1+1
2
>>> a="blue"
>>> print a
blue
```

As you can see, the python interpreter is acting on our input, and displaying output where appropriate. Of course, what we type needs to be valid:

```
>>> call_home
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'call_home' is not defined
```

Variables defined in an interactive session persist only for the duration of that session:

```
>>> myName = "Ed"
>>> print "Hello, "+myName
Hello, Ed
>>> ^D
$ python
Python 2.5.1 (r251:54863, Apr 15 2008, 22:57:26)
[GCC 4.0.1 (Apple Inc. build 5465)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> print "Hello, "+myName
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'myName' is not defined
```

The "^D" on the fourth line represents pressing control-d, and is how we exit the interactive python interpreter. As you can see, the `myName` variable was lost when we exited the interpreter, and was unknown by the new interpreter. This makes the interactive python shell a *great* playground, where you really can't damage much. Despite this, I end up not using the interactive environment very often. There are a few quirks to using it, and it doesn't always model the way a script will run. In essence, I use a text editor and run the program at a

BookEndz® Docking Stations for Apple Computers

DOCKING STATIONS FOR APPLE COMPUTERS

Convert your MacBook Pro®, MacBook®, or PowerBook®
into a desktop in seconds
without misplacing cables or damaging connectors.

- Aluminum Plate helps in cooling of notebook
- Connectors are routed to rear of Dock
- MagSafe® Housing prevents accidental disconnect

17" MacBook® Pro Docking Station

17" PowerBook® G4 Docking Station

15.4" MacBook® Pro Docking Station

15" PowerBook® G4 Docking Station

13.3" MacBook® Docking Station

- Also available in Black
- 5 USB 2.0 compliant ports (4 port powered or unpowered hub, AC/DC adapter included)
- Built in DVI/VGA full size
- Gigabit Ethernet RJ45

Visit our website for latest product announcement www.BookEndzdocks.com

BookEndz®
Manufactured by Olympic Controls
1250 Crispin Drive • Elgin, Illinois 60123 • USA
Phone: 847-742-3566 • Fax: 847-742-5686 • Toll Free: 888-622-1199
E-mail: Sales@BookEndzdocks.com

bash prompt. For now, we can use vi/emacs/pico or whatever you're comfortable with. If you're a hard-line GUI person, but don't have a choice of GUI editor as of yet, the free TextWrangler from BareBones is very highly recommended (http://www.barebones.com/products/textwrangler/).

Hello, World!

It's probably not too surprising that I'd start off with the classic "hello, world" program. It certainly is an easy way to ensure that our basic environment is working and that we know how to code a basic example. In your text editor, enter the following:

```
#!/usr/bin/env python

print "Hello, World!"
```

Save the file as **`hello.py`** when done. If you're strictly using the shell, set the executable bit on the file:

```
$ chmod 770 hello.py
```

and run it:

```
$ ./hello.py
Hello, World!
```

(If you've opted to use TextWrangler, you can skip the chmod/run dance–you can even skip save!–and choose "Run" from the "#!" menu. Output will show up in a new file. If you want to be *really* cool, add a shortcut to Run using System Preferences > Keyboard & Mouse > Keyboard Shortcuts). "Hello, World" is pretty basic, but it illustrates a few things:

Python is installed and working on your machine.
Python can be used as a run-time interpreter.
The most basic structure of a Python program.
You can copy from an article and type correctly.

Python is designed to be simple, with clear syntax. Not AppleScript-like syntax, but consistent and fairly obvious. It tends to be shorter, yet clearer than other scripting languages. Without explanation, look at the following code, and you can surely understand what it does:

```
#!/usr/bin/env python

a = 53
secondNum = 17
message = "The sum is: "

sum1 = a + secondNum

message = message + str(sum1)

print message
```

(This simply prints "The sum is: 70". Yes, it was needlessly long for demonstration purposes).

While a bit contrived, this is also a useful exercise, and shows us a few things about the language:

Identifiers (or, "variables") have no decorators (like a dollar-sign prefix).

Variable names must start with a letter, but the remainder can consist of letters (upper or lowercase), underscores ('_') or digits (0-9).

Variable case matters! "variable1" and "Variable1" are *not* the same variable name.

Python tends to do what you expect. While the plus sign adds two integers in the first occurrence above (a + secondNum), it is *overloaded* to also concatenate strings as seen in its second use above (message + str(sum1)).

What is not obvious from this example is a point I made earlier: Python is a fully object-oriented language, and everything is treated as an object (but don't feel objectified yourself!). Again, variables are identifiers to memory locations; former and current C/asm programmers should understand this implicitly. To demonstrate this, the `id()` function can be used to retrieve the current memory location. This time, I will use the interactive shell. Type `python` in a command shell to enter the Python interpreter. Assign a string to the variable `a`:

```
>>> a = 'this is a string'
```

display it:

```
>>> a
'this is a string'
```

Let's see its memory address:

```
>>> id(a)
445040
```

Now, add on to `a`:

```
>>> a = a+" that is not a bulldog!"
```

and show the address of a again:

```
>>> id(a)
6223136
```

The address is completely different! What happened? The "`a = a+...`" command doesn't simply add on to the existing string in `a`, but throws a away entirely, creating a new variable from the result of the operation on the right hand side (RHS) of the equal sign.

Not having to worry about underlying memory is a hallmark of most scripting languages. Python takes it to another level, though, by handling garbage collection/memory reclamation, allowing you access to those lower level details and handling memory efficiently. Watch this demonstration:

Create a variable:

```
>>> g=5
```

Create a second variable equal to the first:

```
>>> h=g
```

Print the second variable:

```
>>> h
5
```

Find out where they both live in memory:

```
>>> id(g)
8402264
>>> id(h)
8402264
```

However, reassign one of the variables that point to the same memory:

```
>>> h=7
```

…and the former is *not* altered:

```
>>> g
5
>>> id(h)
8402240
```

So, if two variables are equal, they can simply point to the same location in memory. In this example, like the previous example, assigning something to one of the variables creates a new variable, and it will, most likely, receive a new location in memory. While you don't really have to care about *where* Python stores data in memory, this is a core concept in Python, so please ensure that you understand it before moving on.

Data Types

While no declaration of a variable is needed before use, Python is a strongly typed language and, once assigned, a variable's type cannot be changed. As mentioned earlier, everything in Python is an object. Every object has the basic attributes of identity (memory address), type (how to treat what's in that memory address) and value (what is actually stored in that memory address). Certain objects are *immutable*, meaning that their value cannot change once assigned. Conversely, other types are *mutable*, and can change their value. What are these types? Here is a list of basic data types built-in to Python:

Numeric:

Integers: elements from the mathematical set of integers.

Plain Integer (int): range from -2147483648 through 2147483647.

Long Integer: Basically gives an unlimited range, subject to available memory only (virtual included).

Boolean: True and False (behaves like 0 and 1).

Floating Point Numbers: Double-precision floating point numbers.

Complex Numbers: These represent complex numbers as a pair of machine-level double precision floating point numbers.

Sequences: Any data type in the sequence category is a finite ordered set, indexed by non-negative numbers.

Strings: Simply, an array of 8-bit bytes.

Unicode: An array of Unicode code units. In other words, a Unicode string.

Tuples: This type is a true Python-ism, and represents an arbitrary list of objects, formed by comma-separated lists of expressions.

Lists: Mutable; a comma separated list of arbitrary Python objects.

Sets: Like sequences, however, sets are *unordered*, finite sets of unique, immutable objects. This also means that sets cannot be *indexed* like sequences can.

Sets: Mutable unordered sequences.

Frozen Sets: Immutable unordered sequences.

Mappings: Mappings are finite sets of objects, indexed by arbitrary index sets. This type is mutable and *fast*.

Dictionary: This type is a collection of key/data pairs. Also known as a hash table.

Files: Represents an open file. Remembering that everything is Unix is treated as a file, this type includes on-disk files, but also stdin, std out and stderr as well.

Because everything is an object, the list is more expansive then I'm detailing here (for example, classes are a type, also). Also, custom types can be defined and imported into a program, although, custom types are made up of the foundational types. For our introductory purposes, though, this is just the right amount of information.

You've already seen examples of strings and integers in the examples in this article. Since they're the most familiar and natural to most people, I'll continue to focus on those types for now.

Let's Talk about Strings

Strings in Python fall under the category of a sequence. In effect, they are arrays, and can be treated as such. Python has fast and effective string manipulations and idioms that you'll need to be comfortable with before moving on to anything even remotely advanced.

For this portion of the article, you can use the interactive interpreter to gain an appreciation of strings and string manipulation a little faster, and then we'll put it together afterwards. Get into a shell, type `python`, and look for the triple-greater-than prompt. Best way to learn about strings is through examples. Assign "I am a string" to the variable "text":

```
>>> text="I am a string"
```

Display it, to verify:

```
>>> text
'I am a string'
```

Select only the first character from the variable text:

NAME: Seymour MacStuff
EXPO:
January 6-9, 2009
CONFERENCE:
January 5-9, 2009
macworldexpo.com
Macworld®
Conference & Expo
09
MOSCONE CENTER
SAN FRANCISCO, CA

```
>>> text[0]
'I'
```

(remember: we start counting from 0). Selecting part of a sequence is called a *slice* in Python. Slice of character range 2 though 4:

```
>>> text[2:4]
'am'
```

(A slice includes the first character specified up through *but not including* the specified ending character). Display element 5 through the remainder of the sequence:

```
>>> text[5:]
'a string'
```

Leave the start parameter empty to start from the beginning:

```
>>> text[:4]
'I am'
```

A slice can also accept an option third parameter to specify a step:

```
>>> text[::2]
'Ia  tig'
```

(In other words, start at zero, run through the end, giving every second character). Negative values for the step start at the end of the sequence:

```
>>> text[::-1]
'gnirts a ma I'
```

In most other languages, there are dedicated sub-string functions to perform the kind of manipulations you've seen here. Python generalizes this by allowing slices to work for any sequence type. We'll see more of this as we get into other types.

Strings can be concatenated and printed in a few different ways. Let's assign two variables as strings:

```
>>> a="Hello"
>>> b="there"
```

The print command can use a comma, which adds a space character to the output:

```
>>> print a,b
Hello there
```

The plus sign can be used to add strings together:

```
>>> print a+" "+b
Hello there
```

That's the raw basics of strings, with a little left over for next column.

How Python Works

Before we wrap up this column, I want to make it perfectly clear how this is all working and some unique things about Python. First, Python programs are simply text files. That's it; any text editor can be used to create and edit source code for a Python program. These text files typically are named with a .py suffix, but that's not technically necessary, although I'd urge you to get into the habit of doing so. These text files are run through the Python interpreter. Unlike many other scripting languages that read a line of source and then execute it, Python has a trick up its sleeve.

Before running source code, the Python interpreter creates an intermediate byte-compiled version of the program. This machine-independent version of the source is created automatically for you. It is *this* data that is then fed to a Python virtual machine. All of this is done to speed execution. When a byte-code compiled version of your code is created, it is saved alongside the source with a .pyc ("python compiled") extension. (Of course, it will only do this if you have write permissions to the directory with the source. If you don't, no big deal–python will create the byte-code in memory, and drop it on program completion). Here's another speed saving step: if python finds a byte-code compiled .pyc file with a timestamp newer than the source–in other words, the source code hasn't been touched since the .pyc file was created–python simply uses the already compiled .pyc file. For larger programs, this is a huge benefit in startup and run-time speed.

If you've been following along, and you created a "hello.py" program, but notice there's no corresponding .pyc file, you're right! Python only creates byte-code for source that is imported – a topic we'll cover next month. In the meantime, if you want to fool python into creating a byte-code compiled version, we can (although, be aware that you do not need to do this in the real-world, and is for illustration purposes only). Create a new program called `shell.py` in the same directory as hello.py, and type in the following short program:

```
#!/usr/bin/env python

import hello
```

Do the save-chmod dance outlined earlier, and run it. You'll see the familiar "Hello, World!" output. If you now list the directory, though, you should have a new file named `hello.pyc`. This is a complete representation of the source in hello.py, simply pre-compiled for the Python virtual machine. In fact, python will happily run this without the original source. Remove or otherwise rename hello.py. Now, ask python to run the byte-code:

```
$ python hello.pyc
Hello, World!
```

This ability will come in handy in other ways–something we'll cover in future columns.

Fin

While this was a simple, gentle introduction to Python, it presents the right foundation for us to build on. It also should give you enough to fool around with. Consider a bit of homework, just to make you go through the process: write a program that creates two integer variables, "start" and "end" and one string, "text". Have the program print a slice of the string using the variables and a print statement that precedes the string with "The slice is: ".

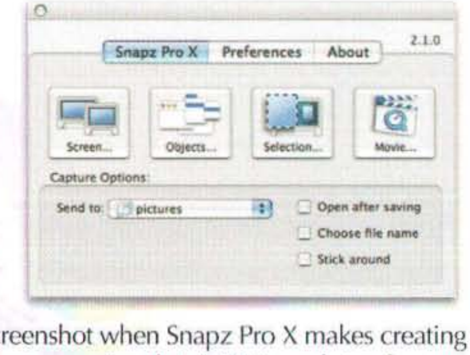
Overall, Python is *easy*, and makes a great first scripting language or introduction to programming. Next month, we'll cover the exercise, and get in a little deeper.

Media of the month: I don't think I've ever had a media selection tie into the column at all. However, for just this once, I'll go for it: "The Complete Monty Python's Flying Circus 16-Ton Megaset" on DVD by Eric Idle, John Cleese, Carol Cleveland, Terry Gilliam, and Terry Jones". Old fan or new, it's just funny.

Until next month, go practice some Python!

About The Author

Ed Marczak is the Executive Editor of MacTech Magazine and the author of "Mac OS X Advanced System Administration v10.5." Offline time is spent with his wife, daughters, rabbit, turtle and fish.


2.1.0
Snapz Pro X
Preferences
About
Screen...
Objects...
Selection...
Movie...

Integrating OS X With OpenLDAP/Samba, Part 1

Configuring Your Mac To Work With Linux Samba and LDAP Servers.

by Noah Gift

Introduction

OS X is an incredible machine, of course, if you are reading this magazine, I didn't need to tell you that. It has arguably, the finest User Interface ever known to man, yet it has a beautiful Unix engine purring under the hood. OS X Server also has a somewhat forward thinking User Interface for managing the raw power of true Unix, as it elegantly hides the real world complexity of dealing with things like Apache and Open LDAP.

While OS X Server is certainly a fine piece of engineering, it might interest many to know, that configuring OS X to talk to a Linux infrastructure is fairly straightforward. I still remember fondly, when parts of OS X had to be altered, in an undocumented way, just to get it to work perfectly with Linux and Unix systems, but many of those problems have disappeared over the years. Virtualization, Leopard, and the further enhancement of Linux, has created a very opportune moment for OS X/Linux relations.

In this article, we explore the basics of getting started with using Samba hosted on a Linux Virtual Machine. Future articles will get deeper into Open LDAP/Samba integration.

Talking Virtually

As I mentioned earlier, virtualization has eased the difficulty of working with Linux. One of the reasons is that, in some respects, the Linux Operating System has become like a document that you open and close. If someone writes a great template for an invoice or resume in Microsoft Word, or iWork Pages, then you simply open the file, and fill in the blanks.

With the current advances in virtualization a Linux operating system can be almost as easy to work with as Microsoft Word. For a recent book I wrote, we released a free, as in beer, Ubuntu 7.10 virtual machine that is pre-configured with OpenLDAP and Samba. You can download it here, start it with VMWare Fusion, and then just play it:

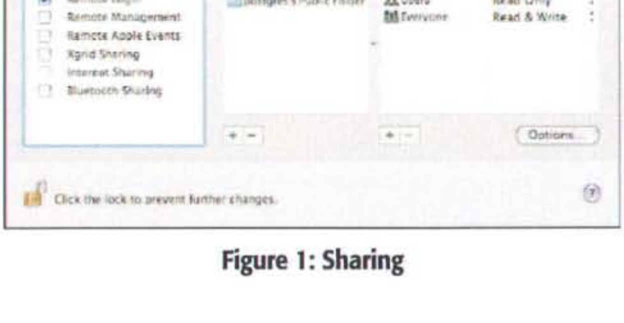
http://examples.oreilly.com/9780596515829/vm/

The username and password for everything on the virtual machine is py4sa, including the Samba service we will use in this article.

Getting To Know Samba

If you are on old Mac guy or gal, you may be more familiar with AFP, or Apple File Protocol, than with Samba. Briefly, Samba is Open Source/Free software that has been around since 1992, and it enables File and Print sharing over the SMB/CIFS protocol. SMB originated with Microsoft as a network sharing protocol for Windows.

Samba is pre-installed on OS X and you can easily create a samba share on your Mac by simply selecting File Sharing under Sharing in System Preferences, as seen in the screenshot:



Computer Name: noah

Computers on your local network can access your computer at noah.local

□ DVD or CD Sharing
□ Screen Sharing
□ File Sharing
□ Printer Sharing
□ Web Sharing
☑ Remote Login
□ Remote Management
□ Remote Apple Events
□ Xgrid Sharing
□ Internet Sharing
□ Bluetooth Sharing

□ File Sharing: Off

File Sharing allows users of other computers to access shared folders on this computer.

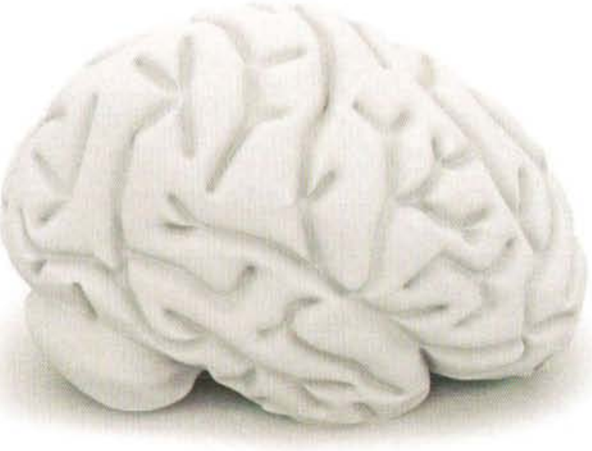
Shared Folders:
ngift's Public Folder
postgres's Public Folder

Users:	
ngift	Read & Write
Users	Read Only
Everyone	Read & Write

Click the lock to prevent further changes.

Figure 1: Sharing

Your email program does not have one of these:



Ours does.

Imagine an email program that was smart enough to observe and learn how you handle your email. Like offering to reply to certain types of email messages using a response you've previously sent. Or giving you the option to auto-file a message you've read based on how you've filed similar messages in the past. What if you could defer a message from your inbox for a few days or weeks, so that it magically reappears later when you're ready to deal with it? What if this program had an amazing junk mail filter that would leave your inbox devoid of spam? What if this program was as familiar and easy to use as Apple Mail?

After three years in development, we've finally created that program. Outspring Mail is the newest email client for Macintosh. We've taken the best features of existing email programs and coupled them with our patent-pending intelligent functions to create a program that will make your email chores far easier. And since this is a current-generation application, it supports major email protocols including POP, IMAP, SMTP and SSL. It also runs natively on the latest Macintosh computers and OS X Leopard. But don't just take our word for it. Check out the details at www.outspring.com.

Outspring Mail. For those who take their email seriously.

Mac

Universal

outspring

5331 Skylane Blvd
Santa Rosa, CA 95403
707-523-7711
http://www.outspring.com

In our case, though, we are going to use virtualization to talk to a share we create inside of the virtual machine we downloaded, so instead of serving the share, our Mac will be the client.

Firing Up A Virtual Machine

If you downloaded the Virtual Machine you should perform the following steps to make sure you get up to speed to follow the rest of the article. If you have problems with the virtual machine networking, or don't want to download a 500MB file, don't worry, because in a later article we go into detail about how to configure this from scratch. Note, you will need to have a copy of VMWare fusion to follow along.

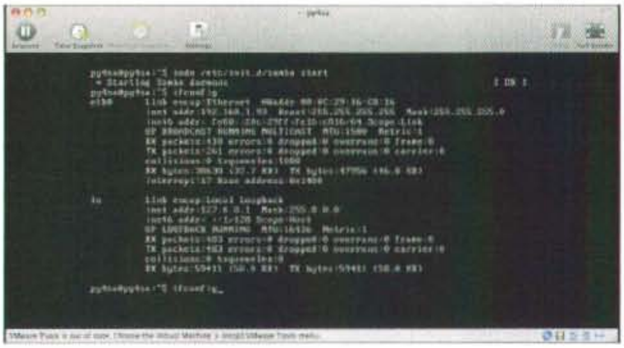
Step 1: Start the Virtual Machine
Step 2: Once running, log in with py4sa as the username and password
Step 3: Double check that samba started by typing at the shell:

```
sudo /etc/init.d/samba
```

Step 4: Ensure Network Is Working by typing at shell ifconfig

```
ifconfig
```

Here is a screenshot of what this looks like a successfully running virtual machine:



```
py4sa@py4sa:~$ sudo /etc/init.d/samba start
 * Starting Samba daemons                                             [ OK ]
py4sa@py4sa:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:36:C8:36
          inet addr:192.168.1.93  Bcast:255.255.255.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe36:c836/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:430 errors:0 dropped:0 overruns:0 frame:0
          TX packets:261 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:38630 (37.7 KB)  TX bytes:47956 (46.8 KB)
          Interrupt:17 Base address:0x1400

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:483 errors:0 dropped:0 overruns:0 frame:0
          TX packets:483 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:59411 (58.0 KB)  TX bytes:59411 (58.0 KB)

py4sa@py4sa:~$ ifconfig_
```

Figure 2: Verifying VM Is Working Properly

Creating A Linux Samba Share On A VM

If you performed the previous steps and everything working, the rest is just as easy. We can simply edit the samba configuration file and add a share for us to connect to. To do this just edit this file: /etc/samba/smb.conf with vim, or your favorite text editor, and add these lines to the bottom:

Listing 1: Adding lines to /etc/samba/smb.conf

```
[share]
    path = /usr
```

Next, just restart the samba service so it rereads the config file by typing in:

```
sudo /etc/init.d/samba restart
```

Now simply browse to the share PY4SA, by creating a new Finder window, as shown in the figure below and authenticating. If you follow along with the steps below you will be able to obtain Read Only Access to the /usr directory on the virtual machine.

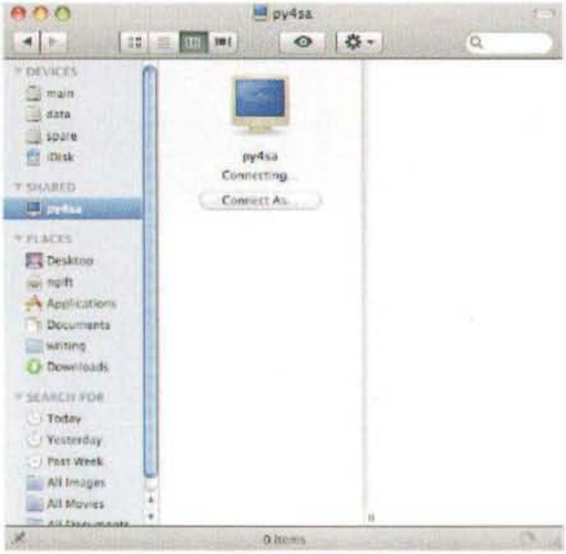
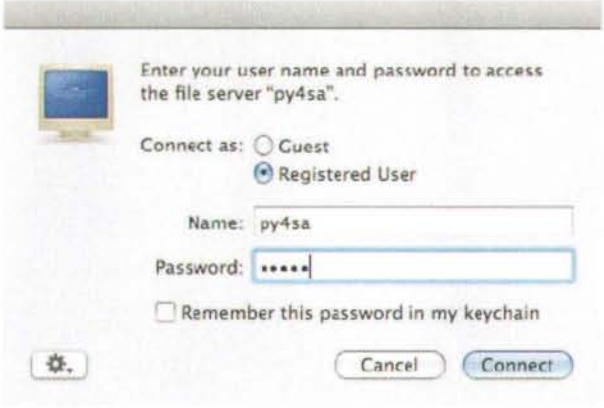


Figure 3: Logging in to a Samba share



Enter your user name and password to access the file server "py4sa".

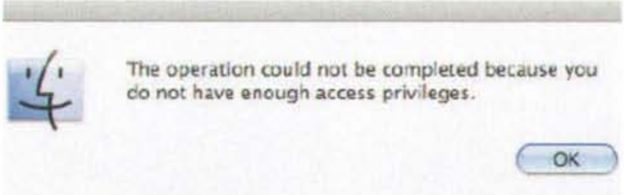
Connect as: ○ Guest
● Registered User

Name: py4sa

Password: •••••

□ Remember this password in my keychain

Figure 4: Authenticating to the share



The operation could not be completed because you do not have enough access privileges.

Figure 5: Read Only Access Trying To Create Directory In Share

Notice, as shown in Figure 5 above, that if you try to create a file onto the share you get an access privileges error. We only have Read Only access! This is because of the extremely basic configuration for [share] we set up. If you are brave, feel free to dig into the smb.conf and fix the problem. Otherwise, next month we go all out on creating a realistic permission structure for an Enterprise or a Home. For now, just getting a virtual server up and running, and then connecting to it is a good stride.

Conclusion

In this article, we began to think about how a virtualized Linux infrastructure might make sense for OS X clients. We created a one way, Read Only connection to an Ubuntu Linux virtual machine via a SMB share we created ourselves.

In the next article we dig in deep and configure LDAP and Samba to work together, create a realistic share permission structure, and authenticate our Mac's directory to a Linux LDAP server, as well as get familiar with managing LDAP through various native and foreign tools.

BIBLIOGRAPHY AND REFERENCES

Noah Gift. "How To Build A Dirt Easy NAS with Samba". Red Hat Magazine,
http://www.redhatmagazine.com/2007/06/26/how-to-build-a-dirt-easy-home-nas-server-using-samba/.
Noah Gift. "Getting Started With Open Directory". O'Reilly.
http://www.macdevcenter.com/pub/a/mac/2007/06/01/discover-the-power-of-open-directory.html
Noah Gift and Jeremy Jones. "Python For Unix and Linux Systems Administration". O'Reilly . ISBN: 0596515820

MT

About The Author

Noah Gift has been a Mac user since his family bought a Macintosh Performa 6300 in 1992, and started connected to BBS networks immediately and then eventually the World Wide Web in 1993 when it become open to the public. He is the co-author of "Python For Unix and Linux System Administration" by O'Reilly, and the upcoming "Google App Engine In Action" by Manning.

Noah has a couple of decades of experience in the Television and Film industry starting off as an editor for ABC Network News as a teenager. He contributed to the first feature animated film for Disney Feature Animation and Sony Imageworks. He also had stints at Turner Studios and Caltech, where he worked for the Nobel Prize winning President as a Mac expert. He has a Master's degree in CIS, and is LPI and ACSA certified. He also organizes PyAtl, the local Python programmers user group in Atlanta.

Currently Noah is consultant, writer and speaker, specializing in OS X/Unix, Linux, Python, and Web development for his company, GiftCS, www.giftcs.com. Many of his projects and writing are available at www.noahgift.com. He can be contacted at noah.gift@giftcs.com

THE ROAD TO CODE

by Dave Dribin

Nobody's Perfect

Document Change Count and Undo

The last session of *The Road to Code* left us with a document-based application that used controllers with Cocoa bindings to hook the UI up to our model. Unfortunately, we have a lingering issue. Typical Mac OS X applications track changes to the user's documents and mark them as "dirty" when he makes changes. Thus, when the user tries to close the window or quit the application, you get a warning sheet about unsaved changes along with the chance to save the document. The red close button in the window bar shows documents that are dirty by placing a dot in the red button, as shown in Figure 1.

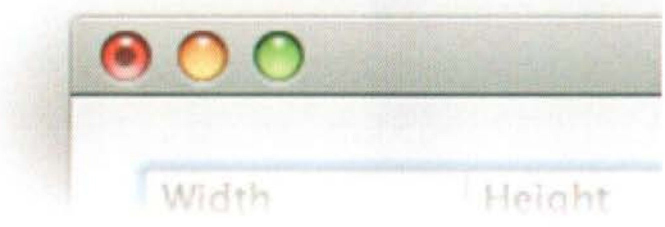


Figure 1: "Dirty " Dot in Close button

Typical Mac OS X applications also allow users to undo and redo their changes. Nobody is perfect, and everyone likes the ability to change his or her mistakes.

Unfortunately, our application does not behave this way. Users are able to change the width and height of existing rectangles or add and remove rectangles, and then close the window or quit the application with nary a "Do you want to save?" warning. On the plus side, Apple's document-based architecture provides hooks for us to manage the state of the document and provide undo support.

The Document Change Count

As a starting point, I have included, in Listing 1 and Listing 2, our `MyDocument` class from last month's article that used Cocoa bindings. Our changes will be centered on this class.

Listing 1: MyDocument.h

```
#import <Cocoa/Cocoa.h>

@interface MyDocument : NSDocument
{
    NSMutableArray * _rectangles;
}

@property (readwrite, copy) NSMutableArray * rectangles;

@end
```

Listing 2: MyDocument.m

```
#import "MyDocument.h"
#import "Rectangle.h"

@implementation MyDocument

@synthesize rectangles = _rectangles;

- (id)init
{
    self = [super init];
    if (self == nil)
        return nil;

    _rectangles = [[NSMutableArray alloc] init];

    return self;
}

- (NSString *)windowNibName
{
    // Override returning the nib file name of the document
    // If you need to use a subclass of NSWindowController
or if your document supports multiple NSWindowControllers,
you should remove this method and override -
makeWindowControllers instead.
    return @"MyDocument";
}

- (void)windowControllerDidLoadNib:(NSWindowController *)
aController
{
    [super windowControllerDidLoadNib:aController];
    // Add any code here that needs to be executed once the
windowController has loaded the document's window.
}

- (NSData *)dataOfType:(NSString *)typeName
                 error:(NSError **)outError
{
    NSData * rectangleData =
    [NSKeyedArchiver archivedDataWithRootObject:_rectangles];
    return rectangleData;
}

- (BOOL)readFromData:(NSData *)data
              ofType:(NSString *)typeName
               error:(NSError **)outError
{
    _rectangles = [NSKeyedUnarchiver
unarchiveObjectWithData:data];
    return YES;
}

@end
```

Before we proceed, we should formally go over some terms. When a document can be safely closed without losing any changes it is called a *clean* document. Once the user starts editing a document, it must be saved or changes will be lost, and the document is called a *dirty* document. The NSDocument keeps track of whether the document is clean or dirty. The isDocumentEdited method returns YES if the document is dirty and NO if the document is clean. If you do nothing it always returns NO, which is why our document can be closed without prompting the user to save. To change the document state, you have to use the updateChangeCount: method. Thus, in your NSDocument subclass, you would have code that looks like this every time the user edits the document:

```
    [self updateChangeCount:NSChangeDone];
```

Why every time? Well, the document's edited state is actually implemented as a count called a *change count*. The document is clean when the change count is zero, and it's dirty when the change count is greater than zero. Why use a change count, rather than just a simple BOOL flag? This is useful once we talk about undo. For example, if the user makes five edits on a clean document, it becomes dirty. But if the user then runs undo five times, the document should be clean again. By implementing the document status as a change count, this is relatively easy: edits increment the change count and undos decrement the change count. We will be covering undo later in this article.

To increment the change count, we pass the NSChangeDone argument to the updateChangeCount: method, as I noted above. So to make sure our document is marked dirty, we just need to increment the change count any time the user makes an edit. Sounds easy, right? Well, it turns out to be relatively easy, but not trivial. Before we start updating the change count, let's list out all the possible ways a user can edit a rectangle document:

Add a new rectangle,
Remove an existing rectangle,
Change the width of an existing rectangle, or
Change the height of an existing rectangle.

Before using Cocoa bindings, these edits went through our document subclass, either through button actions or the table data source. Thus, without bindings, you would update the change count at these places in your document subclass. However, with Cocoa bindings, the array controller now handles all these edits, which actually complicates things a little.

Let's start with marking the document as dirty when adding and removing rectangles. Our MyDocument class has a rectangles property that is of type NSMutableArray. The array controller is bound to this property, so whenever rectangles are added and removed, it uses the setter of this property. Currently, the setter of this property is generated for use with the @synthesize keyword; however, if we write our own setter, we can insert code to update the change count:

```
- (void)setRectangles:(NSMutableArray *)rectangles
{
    if (rectangles == _rectangles)
        return;

    _rectangles = [rectangles copy];
    [self updateChangeCount:NSChangeDone];
}
```

The last line is the important line that marks the document as edited. If you run it now, the document will be marked as dirty as soon as you add a new rectangle.

However, the document will not be marked as dirty if you change the width or height of an existing rectangle. In order to see this incorrect behavior, you will have to save a document, which will mark the document as clean, and then change a rectangle.

To fix this, we need to handle the last two kinds of edits. When a width or height changes, it goes through the array controller, so you may think that subclassing NSArrayController would be the way to update the change count. It turns out that this is not the best way to implement the change count update, as NSArrayController is not meant to be subclassed for this purpose.

Key-Value Observing

Thankfully, there is another way. We briefly talked about key-value observing or KVO in a previous article, but now we are going to use it in earnest. KVO allows one object to observe changes of another object, thus we can use KVO to watch for changes to existing rectangles. KVO is based on key paths. An object starts observing a key path on a target object and gets notified whenever that key path changes. We need to setup MyDocument as an observer for the "width" and "height" key path of all rectangle objects in the _rectangles array. Let's start off by creating a method in MyDocument that sets itself up as an observer to a single rectangle:

```
- (void)startObservingRectangle:(Rectangle *)rectangle
{
    [rectangle addObserver:self
                forKeyPath:@"width"
                   options:0
                   context:&kRectangleEditContext];
    [rectangle addObserver:self
                forKeyPath:@"height"
                   options:0
                   context:&kRectangleEditContext];
}
```

Since each KVO registration is for a single key path, we have to observe each key path independently. We do not need any options, but we are using the context. The kRectangleEditContext is a static variable we need to set at the top of our source file:

```
static NSString * kRectangleEditContext = @"Rectangle Edit";
```

The context is a void * value and is passed in the KVO notifications so you discern between multiple observers.

Remember that `void *` is a pointer to anything. It does not even have to be an Objective-C object, but we are using a pointer to an `NSString *` because it makes it easy to create a unique pointer value. We are using the same context because we want to perform the same action when either the "width" or "height" changes. Next, we have to implement the method that gets called when an observed key path changes:

```
- (void)observeValueForKeyPath:(NSString *)keyPath
                      ofObject:(id)object
                        change:(NSDictionary *)change
                       context:(void *)context
{
    if (context == &kRectangleEditContext)
    {
        [self updateChangeCount:NSChangeDone];
    }
    else
    {
        [super observeValueForKeyPath:keyPath
                             ofObject:object
                               change:change
                              context:context];
    }
}
```

All KVO notifications go through this one method. This is different than `NSNotification`-based notifications, where you can choose a method for notification delivery, and this is why the context is so important. Here, we compare the context against the address of `kRectangleEditContext` we used in to `addObserver:...`. If it is that context, we update the change count to mark the document as dirty. It is important to call your superclass's implementation if you do not handle the KVO notification in case it needs to handle its own KVO notifications.

We also need to create a method to unregister for KVO notifications, which we will use when a rectangle is removed. It's important to always remove yourself as an observer otherwise you may get runtime errors.

```
- (void)stopObservingRectangle:(Rectangle *)rectangle
{
    [rectangle removeObserver:self
                   forKeyPath:@"width"];
    [rectangle removeObserver:self
                   forKeyPath:@"height"];
}
```

We now have our KVO infrastructure in place, but we are not yet observing any rectangles. The place to do this is again in the `setRectangles:` setter. This method gets called with a whole new array of rectangles even if a single rectangle is added or removed. The behavior we want is to stop observing all rectangles in the old array and then start observing all rectangles in the new array. Here's what that looks like:

```
- (void)setRectangles:(NSMutableArray *)rectangles
{
    if (rectangles == _rectangles)
        return;

    for (Rectangle * rectagle in _rectangles)
        [self stopObservingRectangle:rectagle];

    _rectangles = [rectangles copy];

    for (Rectangle * rectagle in _rectangles)
```

```
        [self startObservingRectangle:rectagle];
    [self updateChangeCount:NSChangeDone];
}
```

So, before setting the `_rectangles` instance variable to the new array, we loop through all rectangles and stop observing them. And once we set the `_rectangles` instance variable, we start observing all these rectangles. Finally, we update the change count, as we did earlier, so that adding and removing of rectangles causes the change count to increment.

We have one final detail to cover in our program. If we open a saved document we *still* do not properly mark that the document is dirty after modifying a width or height. The problem is that our `readFromData:…` method is setting the `_rectangles` instance variable directly. This bypasses our `setRectangles:` accessor, so we never start to observe the saved rectangles. The solution is to use the setter by assigning via property dot notation:

```
- (BOOL)readFromData:(NSData *)data
              ofType:(NSString *)typeName
               error:(NSError **)outError
{
    self.rectangles = [NSKeyedUnarchiver
unarchiveObjectWithData:data];
    return YES;
}
```

Remember, using property dot notation is the same as calling the setter method. We're not quite done, yet. The setter marks the document as dirty, which means that a document will be immediately marked as dirty after opening. This is not desired, so the best way is to set the change count to zero after setting the rectangles property:

```
- (BOOL)readFromData:(NSData *)data
              ofType:(NSString *)typeName
               error:(NSError **)outError
{
    self.rectangles = [NSKeyedUnarchiver
unarchiveObjectWithData:data];
    [self updateChangeCount:NSChangeCleared];
    return YES;
}
```

The `NSChangeCleared` argument to `updateChangeCount:` causes the change count to be reset to zero, and thus marking the document as clean.

At this point, our application now properly tracks the document state. If we edit any width or height, it triggers a KVO notification, which in turn updates the change count. If we add or remove a rectangle, it also triggers an update to the change count. Note that saving a document or reverting to saved automatically sets the change count to zero and marks the document as clean.

Just because it works, however, does not mean we cannot improve things a bit. The one issue we are going to fix is the

4.4

fact that the setRecangles: setter is used even if just a single rectangle is added or removed. This is fine if the number of rectangles in our array is small, but it can become a problem, as the array gets larger. While I'm not a big fan of optimizing before profiling (a technique to find slow portions of code), I'm going to show you how you could alleviate the problem if you need to.

Indexed Accessors

Instead of the array controller passing a whole new array every time a rectangle is added or deleted, wouldn't it be nice if it could tell you that a single rectangle has been added or deleted? It turns out that KVC already has the answer for us, and we just need to use it.

Most properties represent a single value, for example the width of a rectangle or the name of a person. Because it is a single value, it is also known as a *to-one relationship*. When a property represents multiple values, as is the case with the "rectangles" property of our MyDocument class, it is known as a *to-many relationship*. This is because for every one MyDocument object, there are many rectangles. For to-one relationships, the standard KVC method naming convention of -<key> and -set<Key> work just fine. For to-many relationships, there are optional methods you can implement for more efficient behavior called *indexed accessors*. On the getter side you have two new indexed getters:

-countOf<Key>
  -objectIn<Key>AtIndex

These two methods allow you to get the number of objects in the to-many relationship or a single object without having to fetch the whole array. Here's how we would implement indexed getters for the "rectangles" property:

```
- (NSUInteger)countOfRectangles
{
    return [_rectangles count];
}

- (Rectangle *)objectInRectanglesAtIndex:(NSUInteger)index
{
    return [_rectangles objectAtIndex:index];
}
```

We implement these using the array's direct access API. While these are nice, it does not really help us with our problem. The are also two indexed accessor methods on the setter side:

-insertObject:in<Key>AtIndex:
  -removeObjectFrom<Key>AtIndex:

Again, to implement these for the "rectangles" property, we would use the array API:

```
- (void)insertObject:(Rectangle *)rectangle
 inRectanglesAtIndex:(NSUInteger)index
{
    [_rectangles insertObject:rectangle atIndex:index];
}

- (void)removeObjectFromRectanglesAtIndex:(NSUInteger)index
{
    [_rectangles removeObjectAtIndex:index];
}
```

The NSArrayController is smart enough to use these methods to add or remove a single rectangle, if they exist. Of course, these methods still do not update the change count, but that's easy to add. We must additionally make sure to start and stop observing these rectangles, as well:

```
- (void)insertObject:(Rectangle *)rectangle
 inRectanglesAtIndex:(NSUInteger)index
{
    [self startObservingRectangle:rectangle];
    [_rectangles insertObject:rectangle atIndex:index];
    [self updateChangeCount:NSChangeDone];
}

- (void)removeObjectFromRectanglesAtIndex:(NSUInteger)index
{
    Rectangle * rectangleToRemove = [_rectangles
objectAtIndex:index];
    [self stopObservingRectangle:rectangleToRemove];
    [_rectangles removeObjectAtIndex:index];
    [self updateChangeCount:NSChangeDone];
}
```

And voilà! With two simple methods, we have optimized setters while keeping the ability to update the change count. We still want to keep the setRectangles: method for replacing all rectangles in one call. This is handy when opening a document, for example.

One nice thing about indexed accessors is that it allows you to implement to-many relationships without using an array internally. While it certainly is easiest to implement indexed accessors with an array, you are by no means required to do so.

Undo and Redo

While changing the document status is a good step towards making our application a standard OS X application, users also expect support for undo. Undo is handled in Cocoa by a class named NSUndoManager. Its purpose is to hold the list of undo and redo actions, and then perform these actions when the user invokes **Undo** or **Redo** from the **Edit** menu.

In order to understand how undo works, we need to dig a bit deeper into Objective-C method dispatching. A few articles back, we talked about selectors and how to use them to send messages. Say, for example, we have a Person class with a method to set the name:

```
- (void)setName:(NSString *)name;
```

You typically call this method using the square bracket syntax with which you are now very familiar:

```
    [person setName:@"Joe"];
```

Remember, the selector is the name of the method, including any colons, in this case "setName:". To call a method with its selector, you would use one of the performSelector: methods provided by NSObject. Thus, an alternate way to call the setName: method by using its selector is:

```
    [person performSelector:@selector(setName:)
                 withObject:@"Joe"];
```

This kind of method calling is called *dynamic dispatch* because the selector and arguments can be changed dynamically at runtime, instead of statically at compile time. As we also discussed earlier how Cocoa uses dynamic dispatch to invoke the actions of controls, such as when a user presses a button. But the nice thing about selectors and dynamic dispatch is that you have all the components you need to have a sort of freeze-dried method call. You can save the target, selector, and arguments in instance variables, for example, and then invoke the method later.

While the `performSelector:` methods provided by `NSObject` are very nice, they do have some limitations. For example, you can only pass up to two arguments and the arguments and return values must be Objective-C objects. You cannot pass primitive arguments or get primitive return values. For that, you have to bring out the big guns: `NSInvocation`.

`NSInvocation` is a class that encapsulates an entire method call, the target object, the selector, and all the arguments. It can also handle primitive types. The downside is that it is a bit of a pain to use. For example, let's say we wanted to set the `width` of a `Rectangle` using `NSInvocation`. Remember that properties also generate setter and getter methods, so we can use the `setWidth:` selector to set the `width` property. Here's the code:

```
    SEL selector = @selector(setWidth:);
    NSMethodSignature * signature =
        [rectangle methodSignatureForSelector:selector];
    NSInvocation * invocation =
        [NSInvocation
invocationWithMethodSignature:signature];
    [invocation setTarget:rectangle];
    [invocation setSelector:selector];
    float newWidth = 30.0;
    [invocation setArgument:&newWidth atIndex:2];
    [invocation invoke];
```

Don't worry about fully understanding this code. Yes, the `newWidth` argument *is* actually the second argument. The first two arguments, 0 and 1, are always set to the target and selector, respectively. I just want to quickly introduce the `NSInvocation` object. If you want to full understand it, please consult the documentation.

The reason I bring up invocations is that `NSUndoManager` uses them under the hood to implement undo and redo actions. In fact, it keeps a stack of invocations for both the undo and redo actions. Say we're starting with a new, clean document and we add a new rectangle and change the width to 30, from the default of 15. Here is the list of actions the user performed, in order:

Add a rectangle at index zero.

Set the width of rectangle at index zero to 30.

In order to undo these operations, we must perform their inverse actions in reverse order:

Set width of rectangle at index zero to 15.

Remove rectangle at index zero.

How do you do you add actions to the undo manager in code? There are two ways. If you have a simple method with only a single argument that is an Objective-C type, you can use selectors:

```
    [undoManager registerUndoWithTarget:person
                                selector:@selector(setName:)
                                  object:@"Joe"];
```

However, if you need to use primitive types, then you need to use the `NSInvocation` variant. Thankfully, `NSUndoManager` allows you to do this without going through the whole rigmarole of creating an `NSInvocation` object. Here's how you would create an undo action to set the width of a rectangle to 15:

```
    [[undoManager prepareWithInvocationTarget:rectangle]
        setWidth:15];
```

Note that even though this looks like it is actually calling `setWidth:15`, it is only creating an invocation for this method. `NSUndoManager` is using some deep Objective-C magic to turn what looks like a real method call into an `NSInvocation`. We don't need to concern ourselves with *how* it does it (it uses `forwardInvocation:` under the covers if you do want to learn more), we just need to know that `prepareWithInvocation:` adds an invocation to the undo stack.

Adding Undo Support

Now that we know how to add actions to the undo manager, we can start modifying our code for undo support. We just need to modify every case where we insert or remove a rectangle or change the width or height of a rectangle. Thankfully, we already went through the process of identifying these locations for dirty document support, and this will help us add undo support. For example, to add an undo action when a rectangle is added, we can use our indexed accessor:

```
- (void)insertObject:(Rectangle *)rectangle
 inRectanglesAtIndex:(NSUInteger)index
{
    NSUndoManager * undoManager = [self undoManager];
    [[undoManager prepareWithInvocationTarget:self]
        removeObjectFromRectanglesAtIndex:index];
    if (![undoManager isUndoing])
        [undoManager setActionName:@"Add Rectangle"];

    [self startObservingRectangle:rectangle];
    [_rectangles insertObject:rectangle atIndex:index];
}
```

This requires a bit of explanation. First, we get an undo manager from `self`. `NSDocument` provides separate undo managers for each document, so each document has their own undo and redo stack.

The next interesting bit is that we can set the name of an action. This shows up in the **Edit** menu to provide the user with more context about a particular undo action. So instead of just showing **Undo** it will display **Undo Add Rectangle**. The only thing to be aware of is that you set the action name *after* adding an action to the undo manager, not before.

And finally, we no longer need to update the change count. The undo manager takes care of managing the change count for us.

We now need to add undo support to the other rectangle setter methods, in a similar fashion:

```
- (void)removeObjectFromRectanglesAtIndex:(NSUInteger)index
{
    Rectangle * rectangleToRemove = [_rectangles
objectAtIndex:index];
    NSUndoManager * undoManager = [self undoManager];
    [[undoManager prepareWithInvocationTarget:self]
        insertObject:rectangleToRemove
 inRectanglesAtIndex:index];
    if (![undoManager isUndoing])
        [undoManager setActionName:@"Remove Rectangle"];

    [self stopObservingRectangle:rectangleToRemove];
    [_rectangles removeObjectAtIndex:index];
}

- (void)setRectangles:(NSMutableArray *)rectangles
{
    if (rectangles == _rectangles)
        return;

    for (Rectangle * rectagle in _rectangles)
        [self stopObservingRectangle:rectagle];

    NSUndoManager * undoManager = [self undoManager];
    [[undoManager prepareWithInvocationTarget:self]
        setRectangles:_rectangles];

    _rectangles = [rectangles copy];

    for (Rectangle * rectagle in _rectangles)
        [self startObservingRectangle:rectagle];
}
```

We basically replaced the change count calls with undo action registration. Earlier in this article, when we reset the change count when the document is loaded. Now, we want to do the same thing, but this time, we want to remove all actions from the undo stack:

```
- (BOOL)readFromData:(NSData *)data
              ofType:(NSString *)typeName
               error:(NSError **)outError
{
    self.rectangles = [NSKeyedUnarchiver
unarchiveObjectWithData:data];
    NSUndoManager * undoManager = [self undoManager];
    [undoManager removeAllActions];
    return YES;
}
```

The last detail is being able to undo individual edits. We used KVO to watch for changes to the width and height of each rectangle, and we can use KVO to support undo as well. The basic plan of attack is to use the KVO notification to add an undo action. The only problem is the KVO notification happens after the key path value has changed. If only we could get the previous value of the key path…

It turns out KVO has this capability. We do have to slightly change the way we start observing objects. Remember when we set the KVO options to 0? We are going to modify that to use the NSKeyValueObservingOptionOld option:

```
- (void)startObservingRectangle:(Rectangle *)rectangle
{
    [rectangle addObserver:self
                forKeyPath:@"width"
                   options:NSKeyValueObservingOptionOld
                   context:&kRectangleEditContext];
    [rectangle addObserver:self
                forKeyPath:@"height"
                   options:NSKeyValueObservingOptionOld
                   context:&kRectangleEditContext];
}
```

This tells KVO to supply the old value with the KVO notification. Our KVO notification callback method can now use that to add an undo action:

```
- (void)changeKeyPath:(NSString *)keyPath
             ofObject:(id)object
              toValue:(id)value
{
    [object setValue:value forKeyPath:keyPath];
}

- (void)observeValueForKeyPath:(NSString *)keyPath
                      ofObject:(id)object
                        change:(NSDictionary *)change
                       context:(void *)context
{
    if (context == &kRectangleEditContext)
    {
        id oldValue = [change
objectForKey:NSKeyValueChangeOldKey];

        NSUndoManager * undoManager = [self undoManager];
        [[undoManager prepareWithInvocationTarget:self]
            changeKeyPath:keyPath ofObject:object
toValue:oldValue];
        [undoManager setActionName:@"Rectangle Edit"];
    }
    else
    {
        [super observeValueForKeyPath:keyPath
                             ofObject:object
                               change:change
                              context:context];
    }
}
```

The change dictionary contains the old value. We use this, plus the keyPath and object to register an undo action. It uses a helper method that sets a value using KVC. KVO, like KVC, will automatically convert primitive number values into NSNumber objects. By using KVC, we also can share the implementation for both the width and height properties.

Conclusion

At this point, you should have an application with full dirty document and undo support. Your users will greatly appreciate the extra effort you've made for this bit of polish. If you are having trouble getting this all working, download the accompanying projects from the MacTech website.

MT

About The Author

Dave Dribin has been writing professional software for over eleven years. After five years programming embedded C in the telecom industry and a brief stint riding the Internet bubble, he decided to venture out on his own. Since 2001, he has been providing independent consulting services, and in 2006, he founded Bit Maki, Inc. Find out more at <http://www.bitmaki.com/> ***and*** <http://www.dribin.org/dave/>.

SecuTech

SecuTech Solution Inc.

UniKey

A Driverless Software Protection Dongle

Easy Solution

Driverless technology reduces customers' support work

Both automatic (envelope without source modification) and APIs protection

Numerous sample codes in various programming languages

Greater Flexibility

Protect software, flash (swf and flv) and video files

Protection in local computers or over a network

Support all popular operating systems, including Windows, Linux, MacOS and Free BSD

Remote update and real time functionality

OEM enables flexibility of case, label, and color

A Cost-effective Choice

Competitive pricing

Experienced and efficient technical support

Life time warranty

SecuTech Solution Inc. | www.eSecuTech.com | info@eSecuTech.com

Code Signing—Get Used to It!

Digitally signed applications and you

by Scott Corley

What Is Code Signing?

So you're a computer. And you're not happy executing just any old software. No, you want to only run software that has been approved by someone you trust. How can you, the computer, tell the good from the bad?

Deciding what software can be allowed to execute is a need that crops up in many places these days. An operating system might want to grant trusted privileges only to drivers that are known to come from a trusted source. A user might want to verify that software they have downloaded from a vendor has not been modified, or that they are not being tricked into downloading something with a misleading name. Or, a cell phone manufacturer might want to retain the power to pre-approve every application that runs on their device, for security or business reasons.

Code Signing is simply a way for a computer to check out an application, a driver, or other chunk of code to verify that it was "signed" by a particular vendor, and that the code has not been modified since it was signed. Code Signing is applied primarily to security and digital rights management efforts.

Overview of Code Signing

Ingredient 1: One-Way Hash Functions

Code Signing requires a one-way hash function and a public-key encryption infrastructure. A one-way hash function simply takes a stream of bytes and calculates a small "digest" that uniquely represents the stream of bytes – similar to a "checksum", but much more secure. A digest is typically 32 bytes or less, a nice, small, unique representation of any arbitrarily sized chunk of bytes. A good one-way hash function can create a unique digest for any sequence of bytes, with a very small probability of a "collision", or any two sequences of bytes generating the same digest. By "very small probability", we're speaking in cryptographic terms, where it would take all of the computers on every planet hundreds of thousands of eons to calculate another sequence of bytes that reduces to the same digest (give or take a few hundred thousand eons, depending on the hash function and whether or not it has recently been discovered to be weak).

Ingredient 2: Public Key Encryption

The second required piece of the code-signing puzzle, public-key encryption, provides a bit of useful magic. I can generate a pair of keys, called my "private" key and my "public" key. My private key, which is always kept secret and accessible only to me, can be used by me to encrypt a chunk of data. Anyone with my public key can then decrypt it. At first, this sounds a bit useless – why would I want to encrypt something that anyone else in the world can decrypt? The great thing is that this data I have encrypted can only be decrypted with MY public key. This is the beauty of digital signing – if I encrypt a 32 byte digest with my private key, you can grab my public key from somewhere and verify whether I did, in fact, "sign" that 32 byte digest with my private key. Nobody else can sign a digest and claim that it was done with my private key.

Digital signatures can be chained together in a very convenient way. In the example above, I have signed a 32 byte digest with my private key, and I am asking you to verify it with my public key. Where do you get my public key? How do you know the entire key pair wasn't created by someone else claiming to be me? In every code signing system, there is a party who holds a "root" private/public key pair. That Certificate Authority is responsible for verifying you are who you say you are. Then, in a bit of digital signature awesomeness, the Certificate Authority uses their root private key to sign YOUR public key along with a brief bit of text explaining who you really are. Thus, anyone who has a trusted copy of the root public key can validate that your public key really belongs to you and that it is "approved" by the Certificate Authority. (In reality, there are also "intermediate" certificates used by the Certificate Authority that can be changed out if needed, for example, in case the intermediate certificate's private key is accidentally disclosed to an unauthorized party).

This public-key infrastructure allows a Certificate Authority to control who can sign code, and it allows end-user's computers to validate a chain of trust. If we combine one-way hash functions with public-key infrastructure used to create digital signatures, we can do a pretty good job of verifying that an application was signed by a party that has obtained a certain level of trust with the central signing authority.

It's always nice to have multiple options.

*HASP, the #1 Choice of Software Developers Worldwide**

**Source: Frost & Sullivan #N1AF-70, IDC Bulletin #34452*

HASP® SRM
SOFTWARE RIGHTS MANAGEMENT

Get the perfect fit! Hardware- and software-based protection, licensing and distribution – all from a single solution!

HASP SRM is a powerful software rights management solution enabling you to secure revenue, maintain competitive advantage and accelerate business growth. HASP SRM gives you a multitude of options for:

- *Strong IP & software anti-piracy protection*
- *Secure & flexible licensing and distribution*
- *Hardware- and software-based protection keys*

Discover how a truly role-based solution reduces the cost of managing the DRM lifecycle. Choose licensing models and protection keys completely independent of the protection process – at any time throughout the software lifecycle. Advanced, but easy to use, HASP SRM delivers enhanced marketing power and takes software rights management to a whole new level!

Request a FREE HASP SRM Developer Kit at www.Aladdin.com/MT8

- NORTH AMERICA: 1-800-562-2543, 847-818-3800
- UK • GERMANY • ISRAEL • BENELUX • FRANCE • SPAIN • ITALY • INDIA • CHINA • JAPAN

Aladdin®
SECURING THE GLOBAL VILLAGE

FROST & SULLIVAN 2007 Global Product Innovation Award

Frost & Sullivan
2007 Product Innovation Award
Software DRM

2008 SIIA //CODiE// FINALIST

SIIA CODiE Awards
2008 Finalist
Software DRM

www.Aladdin.com/HASP

Code Signing Step By Step

Let's say Alice is writing an application for a cell phone, and she is required to digitally sign the application before it can be distributed. In order for her digital signature to be trusted, she must first have a relationship with the entity that is requiring the digital signature (let's call this entity Blueberry, Inc.) Alice first generates a random public/private key pair, using her personal computer, and also generates a Certificate Signing Request that includes her identifying information, such as her name, her company name, her website address, the address of her office, etc.

Blueberry, Inc. is given the public key and the Certificate Signing Request. Alice gets a phone call to verify that she exists, some checks are made on her other information, and she pays a fee. Blueberry has their own public and private key, and they use the private key to sign Alice's public key and Certificate Signing Request in a way that anybody can then verify that Alice's public key does, in fact, belong to her.

Now, Alice can use her private key to sign an application she makes. Instead of using her private key to encrypt the entire application, she simply uses the one-way hash function to create a digest of the application, and then signs the digest with her private key.

Carl, who has a Blueberry cell phone, would like to download Alice's application. But Blueberry, Inc. has declared that their cell phones will only execute code that has been signed by an authorized certificate.

At this point, Carl doesn't need to do anything to verify that the application came from Alice. He doesn't even have to be online for his phone to verify the validity of the application.

Carl's phone will use the same one-way hash function that Alice used to create a digest of the application. It will then look at the signed digest that Alice included with her application. Carl's phone will have a master copy of Blueberry, Inc.'s public key that Blueberry put on the phone when it was made. The phone can check to see if Alice's public key is valid (it was signed by Blueberry's private key, so it will check out ok). The phone can then check to see if the digest that Alice signed matches the digest that the phone computed. If the digests match, then it is reasonable to conclude that the application has not been modified, that it has come from Alice, and that Blueberry has verified that Alice is trusted enough to be signing applications.

An additional step can be added by Blueberry, Inc, if they want to approve each of Alice's applications individually. Blueberry can require that the application signature added by the developer be signed one final time by Blueberry. This way, Alice can't publish a new application on Blueberry's system without Blueberry's final approval.

Terminal Reality

Mac development tools are generally based on BSD command-line tools with nice friendly GUIs on top. XCode, for example, is a really nifty, easy to use GUI that relies on the GCC compiler toolchain.

For code signing, there is a command line tool called "`codesign`". When you set up code signing in XCode (in the

The bulk-free scanner to de-bulk your life.

Meet the world's smallest color duplex scanner designed just for the Mac.

With a footprint that's half the size of a sheet of paper, the Fujitsu ScanSnap S300M will cut your workload—and your desktop clutter—down to size. The ScanSnap S300M has an automatic document feeder that holds up to 10 pages and scans both sides of everything from business cards to legal-size documents at a rate of up to 8 pages per minute. Now you can convert stacks of paperwork from receipts, recipes, even a budding artist's masterpiece, into PDFs with the touch of a button. And the ScanSnap S300M is Leopard compatible, with a choice of AC adapter or portable USB power so you can stay organized no matter where life takes you. Tell us how you'll ScanSnap. You just might win a $100 American Express gift check. Visit us at **http://us.fujitsu.com/scanners/tech**

ScanSnap S510M

ScanSnap S300M • AC adapter and USB power settings • Scans paper sizes from 2 x 2 inches up to Legal length • Automatic document feeder holds up to 10 pages • One button scanning to PDF and JPEG • Scans at speeds up to 8 pages per minute • Automatic color, de-skew, orientation and blank page removal • Bundled with Cardiris™ business card software • Includes Leopard compatibility • Duplex color up to 600dpi • Less than 3.1 lbs. • Only $295

Mac Universal

ScanSnap

Advance EXCHANGE

FUJITSU

THE POSSIBILITIES ARE INFINITE

CDW MacConnection MacMall Insight

"Code Signing" section of the Build settings), you are really just specifying the information that XCode needs to execute codesign. In fact, if you look at the build transcript in the Build Results window after building an app with code signing set up, you'll see the exact command that XCode used to do the code signing.

codesign can also be used to get information about the validity of a signed app. The "—verify" option of codesign will check to see if a signature exists, and whether or not it is valid. If everything checks out ok, codesign –verify will say nothing (or, if you provide a "—verbose" option, it will report "valid on disk"). If there is no signature, it will report "code object is not signed" (or possibly some other error), which basically means it can't find a signature to verify. If there is a signature but something has been modified, codesign will report "a sealed resource is missing or invalid", meaning the hash of what you have does not match the hash of what was originally signed.

If you would like to know some information about who signed an application, you can use the "–details" option of codesign. The –details option can be used with varying levels of verbosity, and will show you who signed the app, along with some details of the application bundle. Look for the lines that start with the word "Authority:" to see what certificates were involved in the signing of the app.

Let's try out codesign with an app signed by Apple:

Listing 1: Verifying a Signed App

codesign —verify

All of Apple's applications shipped with Mac OS X are signed by Apple. You can verify a digital signature using the codesign command.

```
% cd /Applications
% codesign –verify –verbose Mail.app
Mail.app: valid on disk
%
```

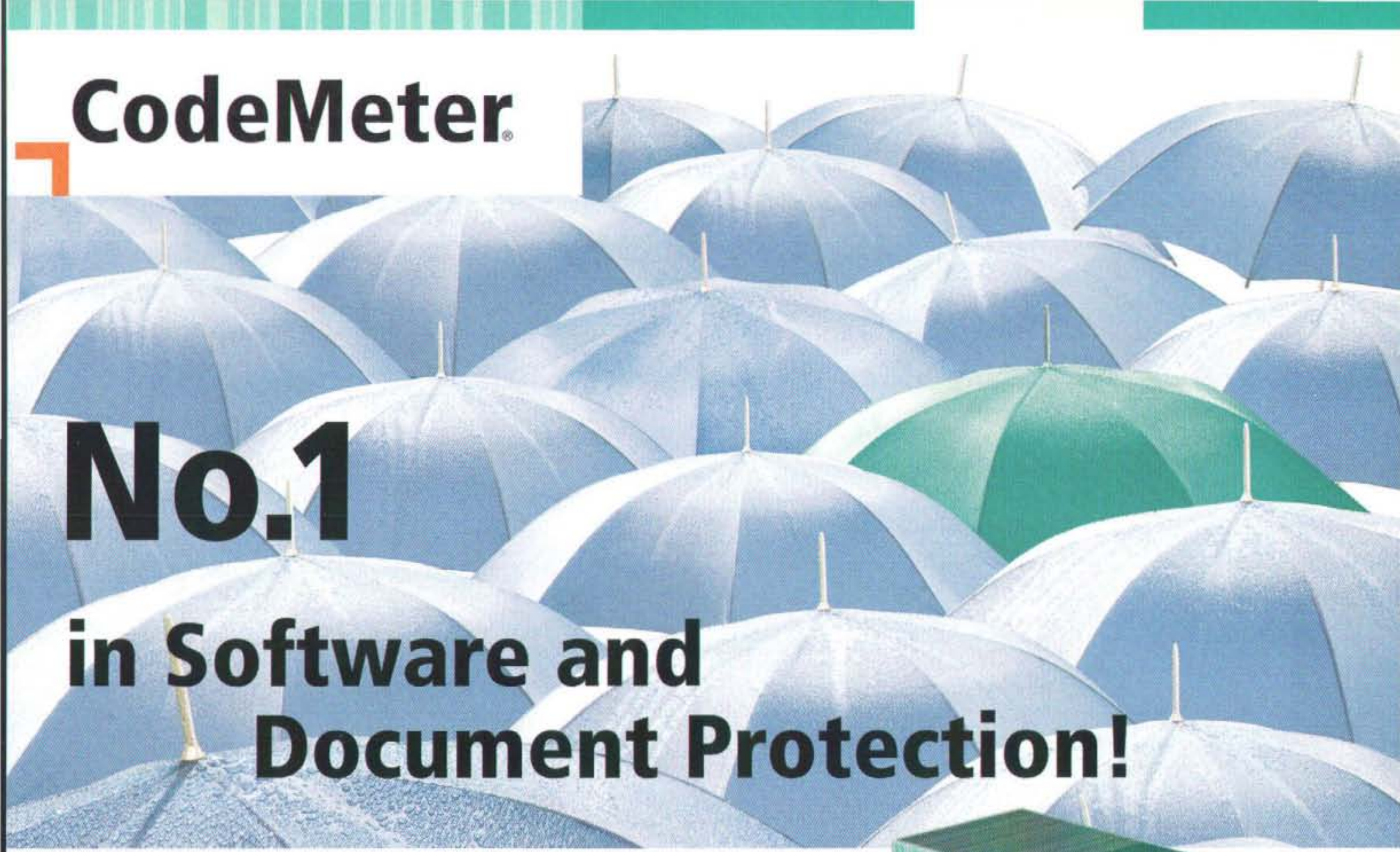
Now let's get some details about a signed app:

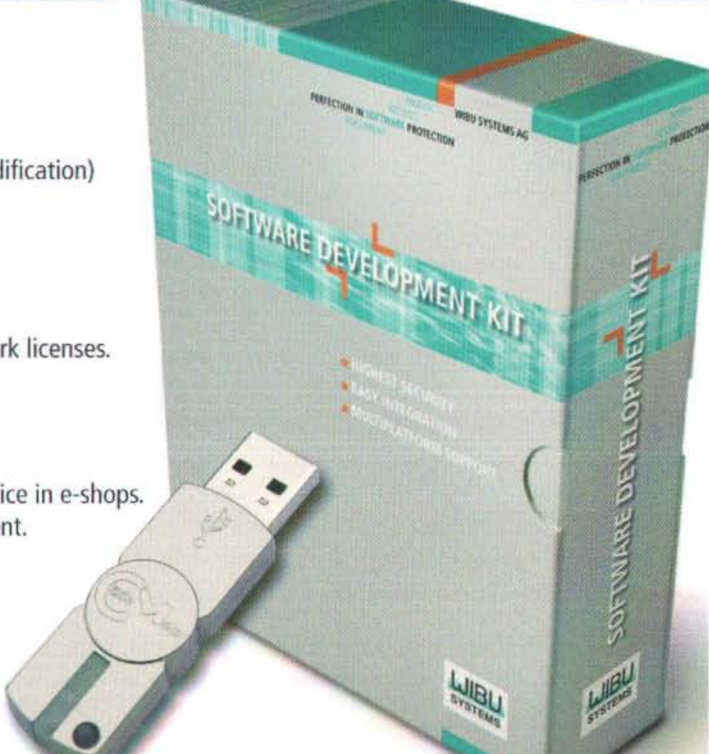
Listing 2: Details of a Signed App

codesign —details

You can get details about who signed an app using codesign. The lines beginning with "Authority" show the chain of certificates that was used to sign the app. This particular app was signed with a certificate called "Software Signing". That certificate has been signed by an intermediate certificate called "Apple Code Signing Certification Authority", and in turn, that intermediate certificate has been signed by "Apple Root CA", the certificate that is at the root of Apple's internal trust chain.

The –details option will not show very much information unless you tell it to be verbose. Here, we have added "-vvv" to kick up the verbosity to get all of the details.




SOFTWARE DEVELOPMENT KIT
WIBU SYSTEMS
WIBU SYSTEMS AG
SOFTWARE DEVELOPMENT KIT

```
% codesign —details -vvv Mail.app
Executable=/Applications/Mail.app/Contents/MacOS/Mail
Identifier=com.apple.mail
Format=bundle with Mach-O universal (i386 ppc7400)
CodeDirectory v=20001 size=14679 flags=0x0(none) hashes=728+3
location=embedded
CDHash=49a4001b87af17822b4839dfb6274f79f3b62f76
Signature size=4064
Authority=Software Signing
Authority=Apple Code Signing Certification Authority
Authority=Apple Root CA
Info.plist entries=27
Sealed Resources rules=9 files=525
Internal requirements count=0 size=12

%
```

Now let's try it with a 3rd party app that is unsigned:
(replace "kdiff3.app" with something you have installed)

Listing 2: Checking the Signature of an Unsigned App

codesign —verify

Apps on Mac OS X do not require a digital signature, and most 3rd party applications will not have one. If you try to verify a digital signature on an unsigned app, codesign will simply tell you that there isn't a signature.

```
% codesign —verify kdiff3.app
kdiff3.app: code object is not signed
%
```

It is also worth mentioning another command line tool, "openssl". We have already discussed "Keychain Access", the very easy-to-use GUI tool included to help create and manage digital signing certificates. If you've ever had to get an SSL certificate or really anything else involving certificates, you've probably run across openssl.

If you ever take a step outside the World of Apple, and need to work with digital certificates, openssl is your friend. It can convert certificates from one format to another, it can create certificate signing requests, and much much more. It does so much, in fact, that I'd recommend not worrying too much about how it works until you need it. If someone needs you to do something with openssl, they'll likely have a step-by-step walkthrough of what you need to do ready to go, and if not, it's usually easy to google up what you need. For example, if you are purchasing a a code signing certificate from a Certificate Authority, they'll have instructions how to use openssl to generate everything they need from you.

And, in super Mac OS X command-line goodness style, openssl is already on your mac. If you want to dig in, just type "man openssl" and read away.

What is a Certificate Authority, Really?

If you want to digitally sign your Mac apps, you'll need to purchase a code signing certificate from a Certificate Authority. They will verify that you are who you say you are and issue you a certificate that is signed by them. This puts you in the "trust chain" that can be verified by the root certificates included in Mac OS X.

Certificate Authorities charge a few hundred dollars per year for code signing certificates. They can hit you up each year because your certificate is issued to you with an expiration date. If you sign code after that expiration date, the signature will not be considered valid. Code that you signed before the expiration date will still be considered validly signed after the expiration date, but you just won't be able to sign anything new until you fork over the renewal fee for your signing certificate.

Mac OS X supports code signing certificates issued from a number of well known Certificate Authorities. Other platforms may require code signing certificates from a single source - for example, you can't purchase an iPhone code signing certificate from anyone but Apple.

Pros, Cons, and Basic Realities of Code Signing

Pros of Code Signing

If we lived back in the glory days where it seemed every programmer could be trusted not to do anything malicious, code signing wouldn't be as widespread as it is now. These days, it is handy to have some way to figure out who wrote the code that is running on your computer, and whether or not that code has been modified.

Operating systems can use code signing to validate its own subsystems. A secure operating system can use signed code to grant special privileges – for example, a signed video driver can be given access directly to a video card, while all other software would not have that access. The ease of validating signed code via the signer's public key makes this approach very appealing, and it is used in Mac OS X as well as other major operating systems to ensure that nobody can distribute fake or altered system software updates.

Cons of Code Signing

Code signing, however, can also be used to prevent users from executing software that they would otherwise choose to execute. Most operating systems allow users to choose to execute unsigned code, but on some platforms, particularly cell phones and video game consoles, code signing is enforced. If an application is not signed by an approved developer, it will not be allowed to run. In the case of cell phones, this enforcement is usually done in the name of security, but in some cases, for example video game consoles like Nintendo's Wii, and Apple's App Store for the iPhone, it is also done for business reasons. We haven't seen this type of enforcement on desktop machines, but we'll see plenty more of it on future cell phones and other systems.

Basic Realities of Code Signing

The reality of code signing is that it is one tool in a toolbox, and it is not invincible. Often, unsigned code can be given full access to a computer with the click of a "run it

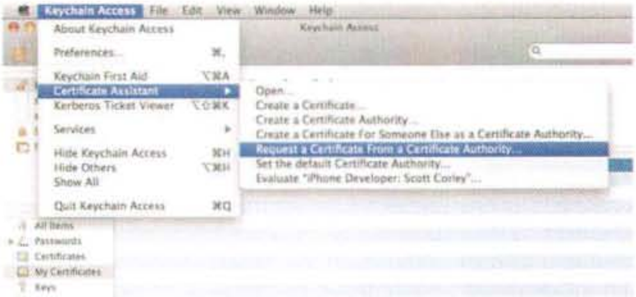
anyway" button that end-users are all too willing to click, or via the entry of a root password that users are all to comfortable entering when asked. Code signatures at a system level are validated by the system itself, so if the system was previously compromised, it could tell itself to allow any code, or code signed by a rogue party, and the end user would never know. And systems that strive to prevent end users from running unsigned code, like cell phones and video game consoles, are constant targets of hackers who go to increasingly astounding lengths to take those restrictions away.

Also, it is important to note that signed code says nothing about what the code actually does. A code signature only assures that the code was signed by a particular chain of certificates. The code itself could still be malicious in some way. If the central signing authority has tens of thousands of "trusted" entities who can sign code, odds are there are going to be varying levels of trust there. It is even possible for a rogue employee to sneak malicious code into an application without their employer's knowledge, then the employer signs the code, and the malicious code makes it out the door with a genuine signature. Because of these situations, code can still be run in a "sandbox" that prevents access to sensitive parts of the operating system and file system. Code that is signed for security reasons, like drivers or system updates, must go through a few layers of trusted internal audit before it gets the final digital signature.

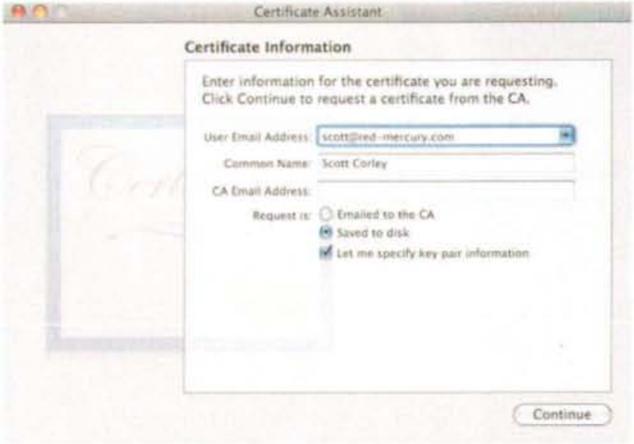
Get Used To It!

Code signing is here to stay. For end users, it usually works behind the scenes, and they'll never even know it's there unless something goes wrong. For developers, however, code signing adds another thing that has to be made to work right, and it has business implications that need to be understood.

The actual code signing is done by tools, and those tools are typically integrated into the developer tool chain, automatically generating the digest and signing it after the build process is complete. But in this type of environment, it is likely that multiple certificates will be involved, depending on whether the application is being built for the developer's device, for beta testers, or for final deployment. Keeping all of these things straight, and verifying that everything is in its right place, can be daunting.



You Can Use Keychain Access Application (in Applications->Utilities) to create a Certificate Signing Request

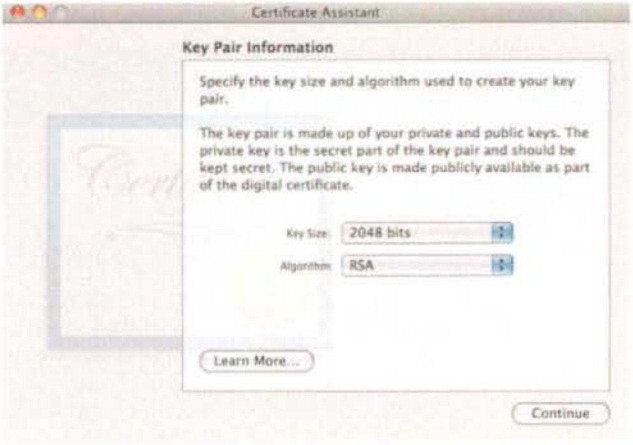


Certificate Information

Enter information for the certificate you are requesting.
Click Continue to request a certificate from the CA.

User Email Address: scott@red-mercury.com
Common Name: Scott Corley
CA Email Address:
Request is: ○ Emailed to the CA
● Saved to disk
☑ Let me specify key pair information

A Certificate Signing Request asks for information about who will be associated with the signing certificate. Typically, the "Common Name" will be a company name, or an individual's real name.



Certificate Assistant

Key Pair Information

Specify the key size and algorithm used to create your key pair.

The key pair is made up of your private and public keys. The private key is the secret part of the key pair and should be kept secret. The public key is made publicly available as part of the digital certificate.

Key Size: 2048 bits

Algorithm: RSA

The Key Size and Algorithm used to generate a key pair must match the standard defined by the code signing authority. 2048-bit RSA keys are common, and the default setting in the Keychain Access program's Certificate Assistant.

Code signing keys can also have business implications. Valid code signing key costs money, so there is added cost which can be significant, particularly if your app is very inexpensive or free. The biggest business implication of code signing, though, is the possibility of not being allowed to have a code signing certificate. Clearly, code signing is as much about allowing trusted entities to sign code as it is about preventing others from signing code. If you think you have a business opportunity on a platform that requires code signing, but for one reason or another you are unable to acquire the appropriate code signing certificate, then you simply won't be able to publish your application on that platform (and perhaps it's time to seek out a partner who can).

Code signing can provide a chain of trust, it can verify the original source of software obtained from untrusted sources, and it can give businesses control over who is allowed to publish software for their systems. For all of these reasons, code signing is here to stay. I hope this has made clear its main uses, how it affects software developers and end users, and how it will become a common part of software development in the future.

MT

About The Author

Scott has been a Mac software developer since he got his first Powerbook in 1991. Scott has written many applications for many platforms, primarily in the video game industry. He currently holds the title of Director at Wideload Games in Chicago, and has developed and published AcidSolitaire Collection for iPhone via his own company, Red Mercury, LLC.

Sample of Word

Display: Sample Text

11072 samples for 69 calls

```
269 Thread_1303
 269 _pthread_body
  269 ReadSchedulerThreadEntryPoint
   269 TSWaitOnSemaphoreCommon
    269 pthread_cond_timedwait_relative_np
     269 semaphore_timedwait_signal_trap
      269 semaphore_timedwait_signal_trap

Total number in stack (recursive counted multiple, when >=5):

Sort by top of stack, same collapsed (when >= 5):
    mach_msg_trap      534
    kevent      269
    semaphore_timedwait_signal_trap      269
    semaphore_wait_signal_trap      269
Sample analysis of process 27742 written to file /dev/stdout
Sampling process 27742 each 10 msecs 300 times
```

GEEK GUIDE

Are you asked for advice all the time? MacTech Geek Guides are here to help you understand the right questions to ask, know the most important issues and terms, and give the right guidance to those that look to you for expertise.

Projectors .... by Dennis Sellers

To project and to serve

Projectors here, projectors there, projectors, projectors everywhere.

Forgive my lame verse, but it pretty well sums up the projector situation. A dozen-plus companies make said devices, which certainly come in handy. But how do you know what projector to buy? Is a projector that's great for a sales presentation a good one for watching movies? Should you buy a portable or stationary model? And what the heck is a lumen anyway? Be patient. MacTech is here to explain it all.

Weighing in

First up, let's look at one of the major factors in picking out a projector: weight. If you're mounting a projector in a home theater setting or stationary office environment, weight won't matter that much. If you're pushing it from office to office or classroom to classroom, weight becomes more important. And if you're a road warrior, lugging the projector all over the country or around the world, you're going to want a small model. The most svelte portable projectors weigh around three pounds.

Resolution

Lots of laptops come with XGA screens these days so, for those on the go, a portable projector makes sense. XGA, or Extended Graphics Array, is an IBM display standard that usually means your laptop supports 1024 x 768 pixel resolution or better. A projector with 1024 x 768 resolution is more than adequate for PowerPoint, Keynote, text presentations and the like. For presentations and general office tasks, there's little need to spring for an SXGA (1280 x 1024) projector. On the other hand, unless money is really tight, don't drop down to the SVGA (800 x 600) models unless your needs go no further than PowerPoint/Keynote slides and simple graphics illustrations. On the other extreme, unless you're into very high-resolution workstation applications, there's little need for an UXGA (1600 x 1200 resolution) projector. Few products on the market have this native resolution, at least as compared to the other resolution options mentioned.

Be the brightest in the room

When you're considering an office projector set-up, you'll need to factor in the size of the room you're presenting in, the screen size onto which you'll be projecting and lighting conditions. The general rule of thumb is to pick a projector that's rated with 1500 lumens or above. A lumen (see, we told you we'd explain) is a unit of measurement of light. ANSI (American National Standard Institution) lumens represents the amount of light emitted by a light source, such as a light bulb or, for our purposes, a projector.

Projector brightness, though one of the most important features considered when buying a projector, is still a somewhat elusive standard. Because the brightness of a projector is largely a function of its lamp, lens, and optical engine, it will vary from projector to projector, even in projectors of the same make and model, or with one projector, depending on the age of its lamp. But as a general rule of them, if you're using a really big screen in a really big room, or are using it for more than 20 people, you should give serious consideration to a projector with at least 2000 lumens.

Connections for the future

When it comes to picking out a projector—however you're going to be using it—you'll have a choice of connectivity options: composite video, S-video, analog computer (or VGA), DVI digital inputs and wireless LAN connectivity. When making your choice, decide which devices you'll be using it with both now and—this is the important part—in the future. A low end projector with a few basic connectivity options may look like a sweet deal now, but that could go sour if you find that, down the road, you need to hook it up to other devices with which it's not compatible. Plan ahead.

Wireless projectors are becoming more and more popular. There are several benefits. You don't have to deal with cables. It's easier to switch between multiple computer sources. And presenters have more flexibility in arranging their set-up.

There are two main types of wireless projector. With "real time" wireless projectors the projector has a constant link to its source over a wireless link. The projector displays the source image in real time. With"upload" wireless projectors, files are uploaded to the projector over a wireless link. The uploaded files are later displayed by the projector so you don't have to have a computer on hand.

Which is better? Here's some advice from the Projector Point web site (http://www.projectorpoint.co.uk): "In most cases, you will find real time wireless projection to be a more useful technology. 'Upload' wireless projectors are only really useful where the same presentation is shown repeatedly using the same projector. Since this advantage is only applicable to a small minority (most users, at some point, want to make last minute changes to their presentation) we would generally recommend real time wireless capability.

" 'Upload' wireless projectors do, however, give you the advantage of being able to project without a PC. For this to be useful, however, you still need to have your PC within range of the projector at some point, whether it's at your office before you go to a presentation, or whether it's at the presentation itself. In the former case, you may as well use a cable. In the latter case, you need to have your PC and projector in the same place anyway, so the advantage is nullified."

All-around good guys

For most of us, multi-use projectors, which are designed to handle work presentations and DVD movie playback at home, will suffice. Reasonably priced, these projectors usually offer a good compromise of weight and features. Look for one with user-selectable screen aspect ratios. Most multi-use projectors come with two user-selectable aspect ratios: 4:3 (standard) and widescreen (16:9). The former is what you'll want to use for computer presentations; the latter is a must-have for letterbox movies.

The higher the contrast ratio, the richer the black appears on the image. If you're using your projector for video purposes, choose one with a high contrast ratio to deliver high quality images. Digital Light Projector (DLP) technology is typically found in multi-use projectors because it offers significantly higher contrast ratios (about 2000:1) than LCD (400:1). As a rule, DLPs have the best light output of the smaller projector types. They have a better contrast ratio so get darker blacks and whiter whites. They have more contrasts and they also provide a more "film-like" image.

It surprises most people, but if you're going to be using your projector simply for viewing movies, a 1000 lumen model will serve just fine. Why? You're most likely going to be viewing a film in a dark or semi-dark room. In this case, a 1000 lumen projector can cast a sharp image up to 100 inches wide with no problem.

Home theater projectors also don't require the maximum resolution (resolution is the number of dots of light that appear on a screen or a projection to make up a projected image in a given space). Lots of movie projectors in the market today are 480p models; they're good enough for DVD movies, TV broadcasts and some game consoles. However, you'll probably want to invest in a higher-resolution projector if you're into Blu-ray or High Definition Television (HDTV) signals, or if you want to "future-proof" your purchase.

Though relatively few projectors come with wide screen support, this is the wave of the future, Ben Joy of InFocus says. High def video conferencing is becoming easy to use and an increasing attractive alternative to travel with the cost of plane tickets and oil, he adds.

Living color

When picking out a projector, you'll want to make sure the color quality is up to your standards. When you see a demonstration of a projector, examine how each model lets you change or enhance colors. Are there adjustments for each of the major color inputs: red, green, and blue? And what about the other colors in the technology palette—cyan, magenta, and yellow? Be sure you can adjust each of these six colors independently to increase the hue of the colors intended, without skewing the other colors. What's more, a saturation adjustment can allow users to simply intensify certain colors more precisely, bringing images into a vivid and vibrant piece of art. Adjustments should be menu-driven and user-friendly so that you can easily get the desired color output quickly.

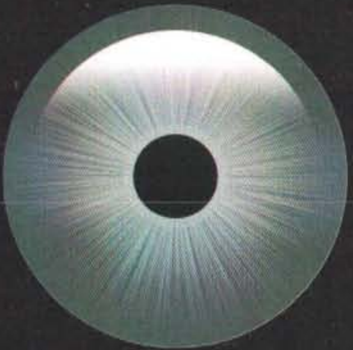
Why is color consistency important? Electronic devices such as projectors, LCD monitors, digital cameras, laptops, and printers usually display colors based on individual manufacturers' presets, says James Chan, director of Projector Product Marketing, Presentation Products Division, Mitstubishi. This means that the same color presented on each device (or over the Internet) can look vastly different than the actual image.

Brightness uniformity is also important. Uniformity is the percentage of brightness carried from corner to corner and edge to edge of your image. A higher uniformity rating means better consistency throughout your image. For the most consistent images, look for a uniformity rating of 85 percent or better.

On the other hand there's such a thing as too much brightness. Too bright a projector in a darkened room can lead to eyestrain. With that caveat be sure to get one that can project an image at the size you'll need and which will still be bright enough to stand up to the ambient light you'll normally use it with.

The roar of the the fans

You'll also want to consider fan noise. The constant humming of a noisy projector can get old quickly. Before purchasing check the Fan Noise rating. Each of the projectors listed in the Projectisle database (at http://www.projectisle.com.au/projector-guide-choosing.asp) have their Decibel ratings listed when published by the manufacturer. The average noise level of today's projectors is



35dB, but if you can spring for a 25dB model do so. They're noticeably more quiet.

Through a glass, brightly

Two other factors to consider when choosing a projector are lamp type and the lenses. The most common types of lamps now used in projectors are UHP (Ultra High Performance) and UHE (Ultra High Efficiency) lamps, although some models still use metal halide, as well. Lamp life will be rated for any projector you buy, and typical lifespans are between 1,000 and 4,000 hours. Replacement bulbs ain't cheap and will generally set you back $250 to $600.

BuyerZone.com says that knowing the replacement cost of the lamp can help you determine your future spending on a given projector. If you're comparing two projectors of the same brightness, ask about the wattage of their respective lamps. If there is a difference, go with the projector that has a lower-powered bulb. Also look for projectors that have an "economy mode."

BuyerZone.com says this setting reduces the power consumption of the projector, cutting brightness by about 20 percent in exchange for less noise, reduced electricity usage, and longer lamp life spans. Some models have a bulb lifespan of 3,000 hours in economy mode, as opposed to 2,000 hours in standard mode.

Zoom lenses are pretty much the standard on modern projectors; you'll want to check how much a lens' f-number (the smaller the number, the more light) changes at different zoom settings. If the change is minimal your image will remain uniformly bright regardless of lens position, notes BuyerZone.com.

If weight isn't an issue, try to find a machine with an all-glass lens (some are made with lightweight plastics). Glass is the optimal filter for projecting images and will give you a clearer picture, but adds a lot of heft to a projector.

Also, LCD projectors have filters that have to be cleaned regularly. DLP projectors have no filters so you don't have to worry about cleaning them.

Projecting your budget

So what can you expect to pay for a projector? Following is a pretty good summary from the Projector Central web site (http://www.projectorcentral.com/home_theater_buyers_guide_6.htm):

- $1,000 or less: Even under $1,000, there are some solid home theater projectors available. The least expensive ones are the 854x480 models (also known as 480p). They display DVD very well, and some (but not all) are capable of delivering very good HDTV also. However, there are some 1280x720 (or 720p) models that have dropped below $1,000 as well. This is the most versatile resolution to get into if you are just starting out and can afford it, as the 720p models are much better with HDTV, HD DVD and Blu-ray than are the 480p models.

AT&T 3G
9:42 AM
SMS
Text
Tuesday
9
Calendar
Photos
Camera
YouTube
Stocks
Maps
73°
Weather
Clock
Calculator
Notes
Settings
iTunes
App Store
Phone
Mail
Safari
iPod

- $1,000-$4,000: If you have anywhere from $1,000 to $4,000 to spend, you're in the price bracket dominated by 720p projectors. Many of the most popular projectors on Projector Central fall into this price bracket, including highly flexible 720p LCD projectors and single-chip DLP projectors with superb contrast.
- $4,000 and up: There are, naturally, lots of high-performance 720p projectors available at or beyond this price, including some three-chip DLP projectors. However, once you cross the $4000 line, the next step up in resolution, 1920x1080, becomes attainable. The new 1080p projectors, when coupled with high-definition signal sources, offer the ultimate in HD home theater, at least for now.

Then, of course, there are plenty of extras and options that may prove useful to your particular situation: Onboard memory card readers for image viewing, security features, wireless connectivity, built-in dual-channel sound speakers, etc.

LCD projector warranties range from one to three years on parts and labor depending on the manufacturer. When offered, standard bulb coverage tends to be for 90 days.

MT

About The Author

Dennis Sellers is a long time journalist. He started in the newspaper business, but has been in the online journalism business for the past 15 years. He's the editor/publisher of Macsimum News (http://www.macsimumnews.com)

Runs On Every Platform

CrashPlan PRO inaugurates a new era in cross-platform backup.

We believe every business has the right to demand:

Platform independence. Choice of platforms: Mac, Windows, Linux, Solaris

Freedom to move. Your laptop is backed up when you need it, wherever you are

Right to restore. Empower your users to recover from their own mistakes

Unencumbered performance. File protection without slowing you down

Unrivaled protection. Onsite, offsite and online backup in one easy to use solution

Now…go exercise your right to a 30-day free trial at www.crashplanpro.com

People Friendly. Enterprise Tough.
www.crashplanpro.com

CRASH**PLAN**PRO™
Continuous Backup for Business.

We're Code 42 Software, and we approve this message

If you want a high-end projector — and can part with around $4,999 (street price), a good choice is the NEC NP3151W, a XGA meeting room projector that boasts 5000 ANSI lumens (and, believe us, that's really bright), a 500:1 contrast ratio and a resolution of 1280 x 800 (native) and 1600 x 1200 (resized). It weighs 16.1 pounds and has wireless and wide format features. The projector has been designed to deliver home-theater level quality for professional environments.

It features WXGA (1280 x 800) native resolution, networking technologies and up to 3000 hours of lamp life (in ECO Mode). Its vertical and horizontal lens shift enables flexible projector placement. The remote desktop connection allows the projector to connect to a networked computer remotely.

As HD video content is more and more frequently displayed in corporate and education settings, the NP3151W has been designed to deliver home theater-level video quality for professional environments. it features advanced HQV video processing, detail enhancement, contrast enhancement, advanced signal noise filtering and technologies designed to eliminate undesirable de-interlacing imperfections.

Additionally, NEC offers a line of optional bayonet lenses allows for quick and easy lens changes that allow a screen size from 40 inches to 500 inches. The NP3151W's lamp saver technologies are designed to protect the projector from overheating. The lamp life cycle is rated at 2000 hours — or 3000 hours in economy mode. For inputs the NP3151W supports Component (3 RCA), S-Video, Composite A/V (3 RCA), DVI, and Analog RGB (15-pin and 5 BNC).

The projector has built-in stereo speakers. There are also three stereo mini audio jacks, a 9-pin serial PC control port, a 15-pin RGB output, and a USB port for a keyboard or mouse. Network connectivity is accomplished by way of an RJ-45 Ethernet port and a USB mounted 802.11 a/b/g Wi-Fi adapter. Remote desktop connection allows the projector to connect to a networked computer remotely.

The projector operates at a 31dBA (economy mode) noise level.

Real World Usage

We put this projector through its paces in some of the toughest environments. In short, it performs very well, has a great, hugely bright picture just as you would expect from a $5000 projector.

Even more amazing is that we used it in some outdoor environments. In one case, we put the projector, along with a portable screen from Da-Lite, in an open tent in the mid-day sun. The tent had three sides to it, with one side open, and even with the sun directly overhead, the picture was quite visible. If you've ever attempted such a thing, you'll realize how impressive this is.

The projector did well in a variety of other indoor and outdoor setups, with brightness and quality of picture being a constant stand out. The sound that came from the unit was quite good, being loud and clear enough to have the projector outside, and yet show a full length movie for a a good size group to hear.

The ability to move a signal over either Ethernet or WiFi was quite a plus for both computer based presentations, or video playback. The only thing that we scratched our head about on this unit was that networking only worked with video, and not audio. So, for example, you could not show a video unless it had no sound, or the sound was handled through a separate sound system. Hopefully, this is something that NEC will enable for the future.

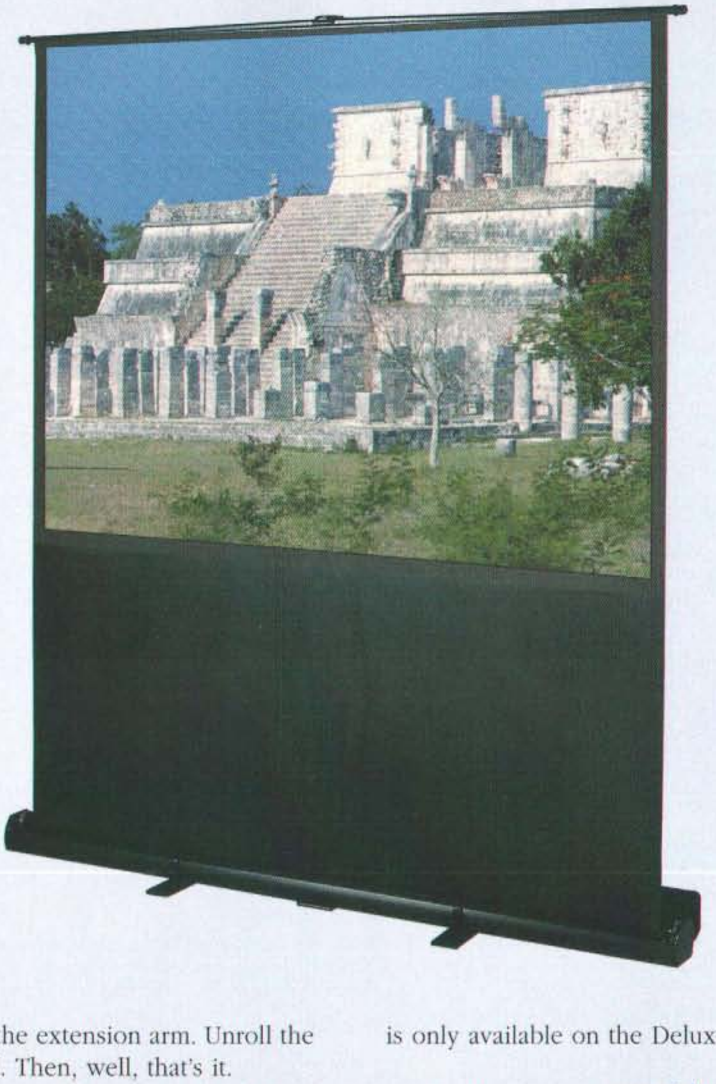
In short, this unit was very "Mac-like" in that it "just worked", and gets a big thumbs up from us here at MacTech.

—Dennis Sellers and Neil Ticktin

REAL WORLD REVIEW: Da-Lite Deluxe Insta-Theater

Many of us watch a movie on our Mac from time to time. Generally, that works out pretty well. But sometimes size does matter, and even if you have a 30-inch Cinema Display, a bigger screen for viewing movies is always a great thing.

One good solution, if it fits your budget, is tricking out your Mac with a projector and a stow-away screen (as well as, optimally, a good sound system). We looked around the market, and found a solid screen choice is the Da-Lite Deluxe Insta-Theater. And, then put it to some real world use. First impression? Wow. Simply wow. Not sure how else to put it.



It's available in 60, 80 and 100 inch diagonal sizes in NTSC Video format and 73 and 90 inch diagonal sizes in HDTV format. List prices range from $490 for the 60-incher (video format) to $740 for 90-inch model (HDTV) with Da-Lite's Wide Power screen surface. However, you can find them for much less if you shop around.

What's Wide Power? According to the folks at Da-Lite, it's a screen surface that combines the features of a higher gain screen surface of 2.2 with a wider 45 degree viewing angle. The smooth acrylic surface can be cleaned with mild soap and water.

The big advantage the Da-Lite screen has over many of its competitors is the ease of setting it up, then storing it when it's not being used. The Deluxe Insta-Theater sports a pneumatic scissors mechanism in an aluminum case for raising and lowering the screen surface. To set it up, release and raise the extension arm. Unroll the screen to the height you need. Then, well, that's it.

The Deluxe Insta-Theater is sturdy as it sits directly on the floor using two thin feet that swing out to 11.5 inches in length. The pneumatic scissors mechanism helps stabilize and balance the screen when it's fully extended, as well as helping ensure the screen surface doesn't warp. That said, if you use it outside, a breeze will catch it easily and you may want to tie it down. Furthermore, if someone walks briskly behind it, you would see their air disruption as you would on any screen.

We used it not only in inside settings, but outside settings (both day and night) and found the screen to be bright, ridiculously easy to set up, and a pleasure to use.

When you're done with the screen, press the center of the handle atop the screen and the device smoothly slides back into its case. Said case is only 4.5 inches around with the stabilizer feet retracted.

Da-Lite also makes a series of black nylon cases for the screens to help protect them during transport. The cases range in price from $52 to $90, depending on the model.

The Da-Lite Deluxe Insta-Theater is lightweight (the heaviest model is around 20 pounds), so it's also good for traveling or business presentations (but watching movies on it is a lot more fun). There's also a Theater-Lite version that has no pneumatic scissor mechanism; it sets up by releasing and raising the extension arm, then unrolling the screen to the preferred height.

The Deluxe Insta-Theater is geared more towards traveling professionals, while the Theater Lite is designed with family use and/or gaming rooms in mind. The Lite version comes in 60 and 80 inches with list prices of $350 and $398, respectively. However, note that Da-Lite's Wide Power is only available on the Deluxe Insta-Theater screen line.

Available from www.dalite.com

—Dennis Sellers and Neil Ticktin

MCX: No Excuses, Now!

New options for managing workstations in OS X Leopard

By Greg Neagle, MacEnterprise.org

MacEnterprise.org
Mac OS X enterprise deployment project

Frequently Asked Questions

There is a certain type of question that pops up over and over again on the MacEnterprise mailing list, the radmind-users list, and other lists, forums, and discussion groups where Mac admins hang out. You'll also hear these questions at the Macworld IT track, and at WWDC. The questions go like this: "How do I manage the screen saver so it always asks for a password?" "How can I prevent the OS from asking the user if they want to use a newly connected FireWire/USB disk as a Time Machine backup destination?" "How do I get the Office 2008 Setup Assistant to not pop up for new users?" Or more generally, "How do I manage the user experience on all my machines?" Nine times out of ten, this question can be answered with "Use MCX!".

MCX

For several years, Apple has had a technology for managing workstations and the user experience, often referred to as MCX, or Managed Client for OS X. To take advantage of MCX to manage users, groups, and computers, an administrator uses Apple's Workgroup Manager utility to edit records in a directory service – typically Open Directory, but sometimes Active Directory or third-party LDAPv3 with Apple's schema extensions.

If your organization did not use Open Directory as its central directory service, and was unwilling or unable to extend the schema on its existing directory service, it was difficult to get the benefits of using MCX to manage your Macs and their users.

There were a few options: one, often referred to as the "Magic Triangle", involved binding client Macs to both the organization's central directory services, and to a locally-managed Open Directory server. Mac clients would then receive their user and group info from the central directory, and their client management data from Open Directory.

Another option was for admins to write scripts that replicated some of the behavior provided by MCX – usually by modifying plist files with the defaults command or PlistBuddy.

Far less frequently used was the option to store MCX data in the client's local directory service. This was possible with NetInfo, but the trouble was replicating those settings across multiple machines. You could not simply replicate the NetInfo database across multiple machines; you had to find a way to export the MCX data (and the objects it was attached to), and import this data on other machines.

So when a Mac administrator asked "How do I manage…", and you replied "Use MCX!", they'd often object: "But I don't have an Open Directory server, and my Active Directory admins won't let us extend the schema! So is there another way…?" Using MCX data in the local NetInfo was too difficult, so they spent a lot of time writing scripts to manage things, and then pushed those scripts out to every machine.

Leopard changes the equation. Now there is really no excuse at all to not use MCX to manage your machines. If you don't have a central MCX-friendly directory service, you can store the MCX data in the local directory service. More importantly, since this data is stored as simple files, replicating this to other machines is as simple as copying a few files. If you manage multiple OS X machines, you must have a way to copy files to each machine – that might be a software distribution mechanism like Casper or FileWave, a filesystem management utility like radmind, or even something as basic as Apple Remote Desktop or the `scp` utility.

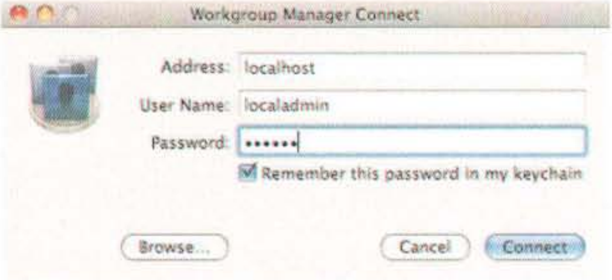
This method also allows administrators to ease into MCX management: you do not need an Open Directory server or extended schema to get started. Instead, you can start with the local directory service. Once the powers-that-be in your organization can see the benefits of MCX, they may be more inclined to invest in the resources needed to set up a "Magic Triangle" or extend the schema on your existing directory service.

Demo Time

Let's demonstrate what can be done with MCX and the local directory service.

You'll need Workgroup Manager, which is part of Apple's Server Admin Tools 10.5. Get them from your Leopard Server install media, or search Apple's website for "Server Admin Tools".

To work with the local directory service, launch Workgroup Manager on a OS X client machine. When presented with the dialog to connect to a server, type "localhost" as the server name, and enter the name and password of a local admin for the local machine.



Workgroup Manager Connect

Address: localhost

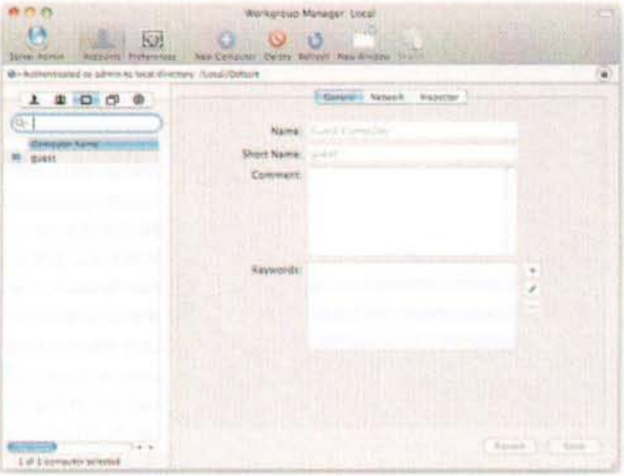
User Name: localadmin

Password: ••••••

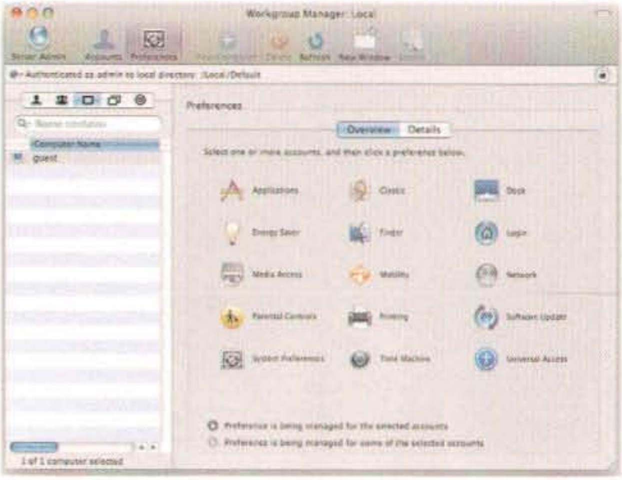
☑ Remember this password in my keychain

You'll see a warning that you are working in a directory node that is not visible to the network. Check Do not show this warning again if you wish, and click OK to dismiss the panel.

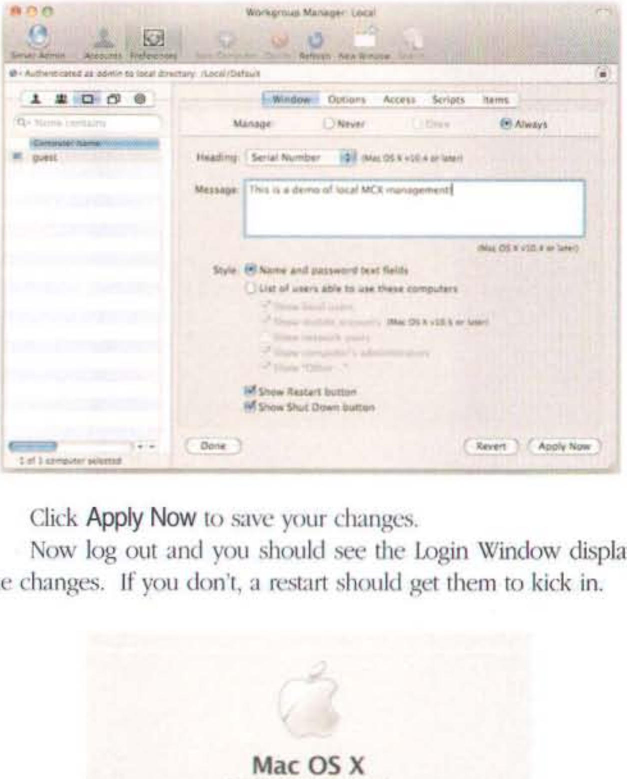
For purposes of this demo, we'll manage aspects of the local machine using the guest computer object. Settings for this object apply to all computers that don't have an explicit computer account record in the directory, which makes it work well for this demo. Choose Create Guest Computer from the Server menu in Workgroup Manager. You'll now have a guest object in the Computer view:



Select the guest computer, then click the Preferences icon in the toolbar. We're going to set some options for the Login Window, so click the Login icon in the Preferences overview.



Under the Window tab, click Manage: Always, then make some changes to the managed settings. Below, I've changed the Heading to display the serial number instead of the machine name, added a message to the Login Window, and changed the Style to show only name and password fields (instead of the default list of users).



Manage: ○ Never ○ Once ● Always

Heading: Serial Number (Mac OS X v10.4 or later)

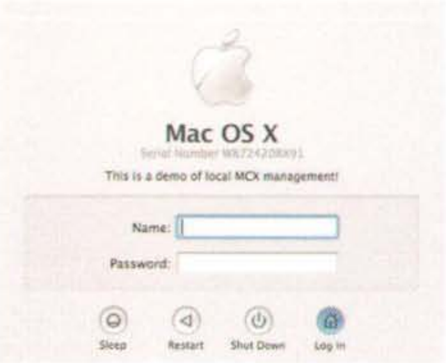
Message: This is a demo of local MCX management! (Mac OS X v10.4 or later)

Style: ● Name and password text fields
○ List of users able to use these computers

☑ Show Restart button
☑ Show Shut Down button

Click Apply Now to save your changes.

Now log out and you should see the Login Window display the changes. If you don't, a restart should get them to kick in.



Mac OS X
Serial Number W8724208X91
This is a demo of local MCX management!

Name:
Password:

The only true graphical terminal server for Mac OS X

Aqua Connect Terminal Server is the only scalable enterprise grade solution that allows the Mac OS X platform to be deployed to multiple diverse devices, including Macs, PCs, and thin clients. Aqua Connect offers a feature set that truly simplifies OS X administration.

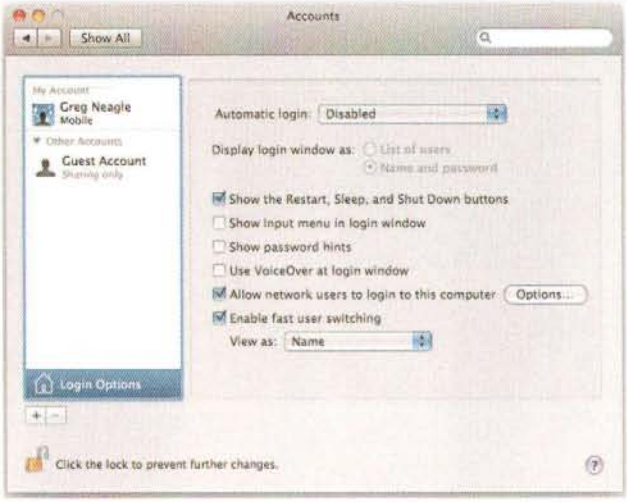
Mac

Universal

Aqua Connect

www.aquaconnect.net
(866) 543-AQUA (2782)

Even more interesting: log back in and open **System Preferences**, select the **Accounts** preference pane, and choose **Login Options**. If you followed my example and set the Login Window to show name and password text fields, you'll see that option set in the preference pane, and grayed out so you cannot change it.



Automatic login: Disabled

Display login window as: ○ List of users ● Name and password

☑ Show the Restart, Sleep, and Shut Down buttons

☐ Show Input menu in login window

☐ Show password hints

☐ Use VoiceOver at login window

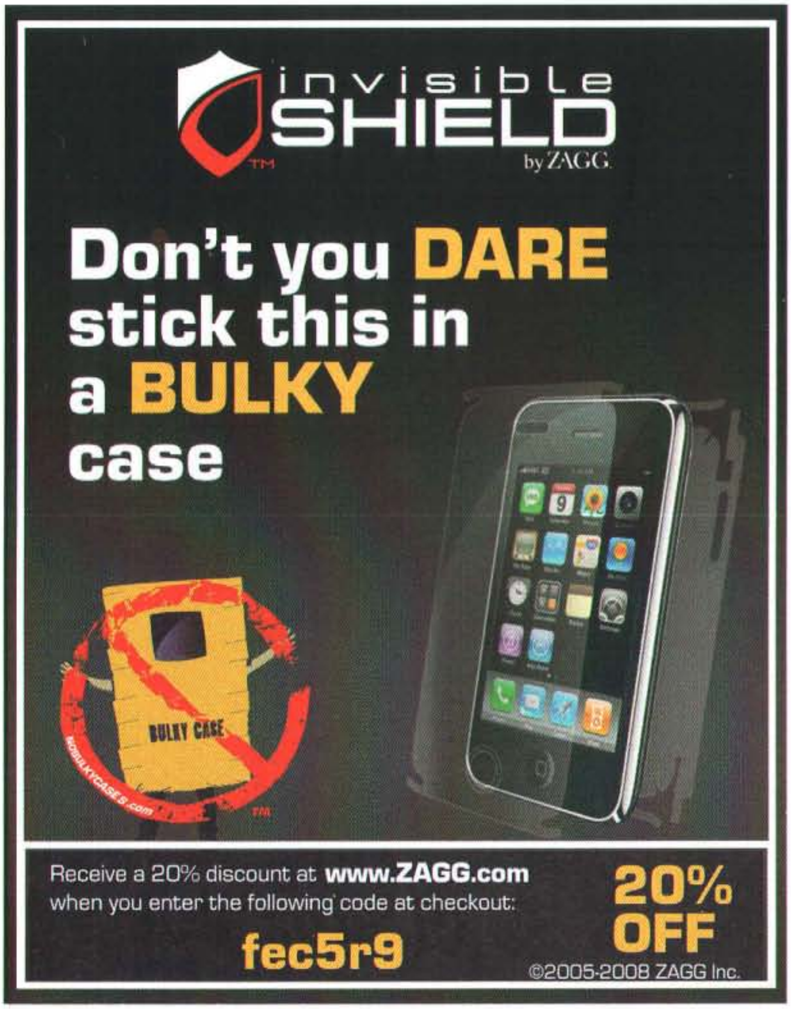
☑ Allow network users to login to this computer

☑ Enable fast user switching

View as: Name

Click the lock to prevent further changes.

This is a huge advantage of using MCX instead of scripts that write to various plists – in many cases, the OS updates the user interface to reflect your management settings.



We've used Workgroup Manager to manage certain preferences for this machine, and stored the MCX record in the local directory service. But what exactly does that mean? To find out, login as an admin and open the Terminal application. You'll need root privileges, so type `sudo -s` and press return, entering your own password when prompted (your account will typically need to be admin level to work. If not, login with an admin-level account).

Now change to the local directory service directory, and list its contents:

```
root# cd /private/var/db/dslocal/nodes/Default
root# ls
aliases         computergroups config          machines
users
computer_lists computers       groups          networks
```

The guest computer object we created, since it's a computer object, is stored in the "computers" directory:

```
root# cd computers
root# ls
guest.plist
```

Let's examine guest.plist:

```
root# cat guest.plist
```

And you'll see a standard OS X plist, which is too long and boring to list here. But you don't really need to deal with the internal structure at all – to replicate these MCX settings on another machine, you need only copy this file to the same location on another machine (and most likely restart the other machine, or restart DirectoryService to get it to notice your changes). If you have a way to push out files to your managed machines, you can now push out MCX settings the same way.

Future Directions

The demonstration isn't very flexible: since all the managed settings are stored in guest.plist, it's hard to mix and match settings. Next time, we'll look at some MCX management strategies using Leopard's new ComputerGroups that allow you to mix and match management policies.

MT

About The Author

Greg Neagle is a member of the steering committee of the Mac OS X Enterprise Project (macenterprise.org) and is a senior systems engineer at a large animation studio. Greg has been working with the Mac since 1984, and with OS X since its release. He can be reached at gregneagle@mac.com.

Achieve post-production image quality during pre-shoot setups. With NEC's full line of professional-grade desktop LCD displays, the guessing games are over. Accurate, consistent and repeatable color performance, along with calibration and sharpness tools, give you confidence in knowing that your on-screen images will look just as bold, bright and beautiful as they will on next month's magazine cover.

Choose from 19" - 30" sizes with widescreen and cabinet color options.

True brilliance. True NEC.

26" NEC MultiSync LCD2690WUXi with SpectraView,™ Color Calibration Kit and 30" NEC MultiSync LCD3090WQXi

www.necdisplay.com

NEC

Advanced CVS

Get more out of your CVS Setup

by José R.C. Cruz

Introduction

In the article "Version Control on a Budget", we learned how to use the CVS tool to manage our project files. Today, we will explore some of the tool's advanced features.

First, we will configure the tool on either the client or server side. Then we will use the tool to keep track of project activity. Finally, we use the tool to manage the project repository.

Be aware that this article assumes that you have a basic understanding of CVS and bash.

Configuring CVS

You can configure your CVS setup in two ways. One way is to add an invisible file in your home directory. This file, called a *run-control file*, tells the tool how to handle each SCM transaction. Users can have their own unique run-control files, or they can share a common set of files.

Another way is to change one of the files in the `CVSROOT` directory. To access those files, type the following statement at the Terminal prompt.

```
    cvs checkout CVSROOT
```

CVS then copies the `CVSROOT` files and stores them on the current work directory. After you made your changes, commit them back to the repository as follows.

```
    cvs commit -m "Made changes to the CVSROOT files"
CVSROOT
```

First, CVS updates its `CVSROOT` directory with all your changes. It then applies your changes to all future commands. If you, however, changed a single file, e.g. `modules`, you can commit just that file as follows.

```
    cvs commit -m "Made changes to the CVSROOT file
'modules'" modules
```

Backing up CVSROOT

Before changing any of the `CVSROOT` files, make sure to create a backup of those files. This will protect your CVS setup from any errors caused by your changes. One way to backup `CVSROOT` is to use the tar tool.

Assume, for example, that your `CVSROOT` directory is stored in `/Volumes/Projects/Database`. Assume also that you want to store the tarball in `/Volumes/Backup`. To create the `CVSROOT` tarball, type the following at the Terminal prompt.

```
    tar -create -verbose -gzip -file
/Volumes/Backups/CVSROOT.tar.gz \
        /Volumes/Database/CVSROOT/*
```

First, the tar tool copies all the files in the `CVSROOT` directory. It then stores the copies in the tarball `CVSROOT.tar.gz`. It also compresses the tarball with the gzip tool.

Now if you made a mistake configuring your CVS setup, type the following statement at the Terminal prompt.

```
    tar -extract -gzip -file
/Volumes/Backups/CVSROOT.tar.gz \
        -directory /
```

The tar tool then updates your `CVSROOT` directory with the contents of the tarball `CVSROOT.tar.gz`.

The .cvsignore **file**

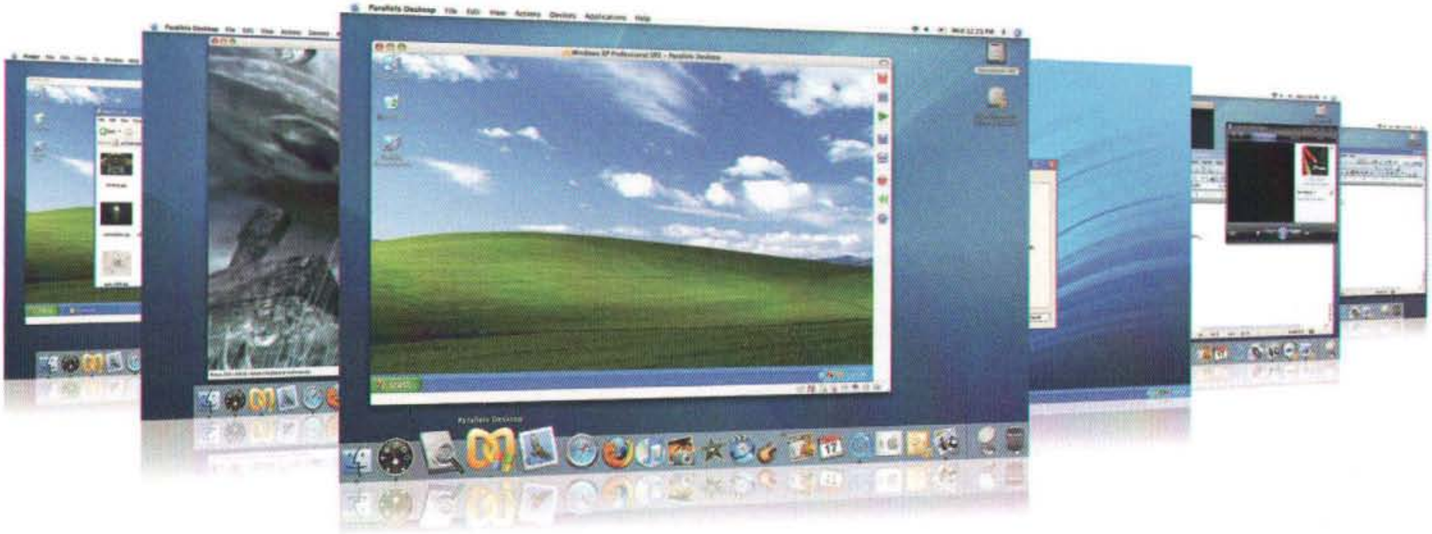
Use the `.cvsignore` file to tell CVS which project file to exclude from its repository. CVS checks this file each time it gets an `import` or an `update` subcommand. Use your favorite text editor to create this file on your home directory. Make sure to precede the file's name with a period, `'.'`.

Each line in the `.cvsignore` file is either a *filename* or a *glob pattern*. Glob patterns work in the same way as regex patterns, but they have a simpler syntax.

Listing 1 is a sample of entries in a `.cvsignore` file. The first entry matches the invisible file `.DS_Store`. The second matches any file with a `.dbug` extension. And the third matches any HTML files that have the name `foobar` or `foobur`.

Plays well with others.

Run Windows and Mac OS X at the same time - without rebooting.



With Parallels® Desktop 3.0 for Mac, you can:

- Automatically open Windows files with Mac software and Mac files with Windows software
- Bulletproof your virtual machine from Windows system crashes and malware
- Run selected Windows-only 3D games and applications with 3D graphics support
- Browse through Windows files and folders without launching Windows
- Move your existing Windows PC to a Mac without losing data or reinstalling any software
- Completely share files and folders between Windows and Mac

Parallels™

Parallels® Desktop 3.0 for Mac provides even more ground breaking features and capabilities than previous versions. Already The #1 Selling Mac System Utility according to NPD Techworld (09/06 – 12/07) and recently named an InfoWorld Magazine "2008 Technology of the Year", Parallels® Desktop is trusted by over 800,000 users world wide.

Want better Windows and Mac integration? Want to run the hottest PC games and graphics software? Worried about security and system mishaps? Parallels® Desktop 3.0 for Mac includes 50+ new features and enhancements, including a number of integration features and virtual machine utilities unavailable anywhere else.

To discover why Parallels® Desktop is the leading desktop virtualization solution for running Windows on a Mac, visit us online or call us at 1 (425) 282-6405.

PC WORLD 100 BEST PRODUCTS of 2007

Macworld BEST OF SHOW 2007

Download a trial today at www.parallels.com/download

Website – www.parallels.com Email – sales@parallels.com Phone – 1 (425) 282-6405

Listing 1. Example of a .cvsignore file.

```
.DS_Store
*.dbug
foob[au]r.htm
```

Before you install your `.cvsignore` file, be aware that CVS already excludes certain file types by default. A sample of these file types are as follows.

```
*.a     *.bak  *.BAK  *.exe  *.o    *.obj  *.old
*.orig
```

For a complete list of excluded file types, consult the online CVS manual (*see reference*).

Finally, you can tell CVS to disable your `.cvsignore` file for certain tasks. To do so, add the `-I !` option to your `import` or `update` subcommand (note the `'!'` character). This option disables not only your `.cvsignore` file but also the default list of file types that CVS ignores.

The `.cvsrc` file

Use the `.cvsrc` file to set the default options for either the CVS tool or any of its subcommands. Like the `.cvsignore` file, the `.cvsrc` file also goes into your home directory. Use your favorite text editor to create this file.

Each line in the `.cvsrc` file has the following syntax.

```
cvs | subcommand option [option]
```

If the leftmost item is a *subcommand*, the `option` argument is one of the option flags used only by that subcommand. But if the leftmost item is CVS itself, the `option` argument is one of the global options available to that tool. Those options apply to all CVS transactions. Read the online CVS manual (*see reference*) for a list of available options.

Listing 2 is a sample of entries in a `.cvsrc` file. In the first line, the `-r` option tells CVS to set the states of the project files as read-only. The `-q` option tells it to keep its feedback messages to the bare minimum.

In the second line, the `-f` option tells CVS to allow committals even if none of the project files have changed. In the third line, the `-f` option tells CVS to always export the head revision. The `-P` option tells the tool to remove any empty directories from the exported project.

Listing 2. Example of a .cvsrc file.

```
cvs -r -q
commit -f
export -f -P
```

Any changes you made to the `.cvsrc` file are used by the next CVS command. You can, however, tell CVS to ignore the `.cvsrc` file for certain tasks. To do so, use the `-f` global option.

Assume, for example, you are using the `.cvsrc` file shown in Listing 2. To commit only those files that you have changed in project `foobar`, type the following statement.

```
cvs -f commit foobar
```

Do not add the `-f` global option to the `.cvsrc` file. This is an unsupported setting and can result in aberrant behavior.

The cvswrappers file

Use the `cvswrappers` file to tell CVS which project files are text files and which are binary files. Use it also to tell CVS how to store each file type in the repository. There are two versions of this file: one in the `CVSROOT` *directory*, the other in the *home directory*. The name for the home directory version is `.cvswrappers`.

Each line in the `cvswrappers` file has the following syntax.

```
file_name | glob_pattern  option mode [option mode]
```

The line starts with a filename or a glob pattern. The `option` argument sets the *storage option* for that file, the `mode` argument the *storage mode*. Each line can have at most two sets of these arguments.

For a *binary file*, e.g. TIFF, add the following entry to the `cvswrappers` file.

```
    *.{tif,tiff} -k 'b'    -m COPY
```

The above entry matches any file that has a `.tif` or `.tiff` extension. It tells CVS not to convert any end-of-line characters in the file. It also tells CVS to always store a copy of that file intact in the repository.

For a *non-mergeable text file*, e.g. a UTF-16 file, use the same option and modes as the binary file. This is to compensate for CVS's lack of support for Unicode-based files.

For a *mergeable text file*, e.g. C-headers and sources, add the following entry to the `cvswrappers` file.

```
    *.{c,h}     -m MERGE
```

The above entry matches any file that has a `.c` or `.h` extension. If the repository has a copy of that file, CVS stores *only the differences* between the two. Otherwise, CVS will store an intact copy of the text file.

Now if you have two versions of the `cvswrappers` file present, CVS will use the entries from both files as one. If both versions of the file have entries for the same filename or glob pattern, CVS will use the entry from the `CVSROOT/cvswrappers`.

Finally, you can override the entries in both `cvswrappers` files by adding a `-k` option to your CVS statement. Read the online CVS manual (*see reference*) to learn which subcommand uses the `-k` option.

The modules file

Use the `modules` file to set shortcuts to specific projects or items in the repository. Shortcuts reduce the amount of typing needed. They are also easier to remember if chosen correctly.

Each line in the `modules` file has the following syntax.

```
shortcut    [options]  repository_path
```

The `options` argument defines how the shortcut will work. It has one of two possible values: `-a` for an alias and `-d` for a directory. The `repository_path` argument is the path to the item or project in the repository.

Listing 3 is a sample of shortcuts in a `modules` file. The first shortcut tells CVS to retrieve the items in project `foobar`. Also, CVS saves those items under the directory `MyFoo`.

The second shortcut tells CVS to retrieve only the items in the graphics directory of project `foobar`. The `-a` option then tells the tool to save those items under the same directory of `foobar/graphics`.

The third shortcut tells CVS to retrieve the items in project `foobar`. But the `'!'` token tells it to exclude those items in the directory `foobar/graphics`. Again, the `-a` option tells the tool to save the retrieved items in the directory `foobar`.

The fourth shortcut tells CVS to retrieve the project items referred to by the shortcut `NoPics`. Note the ampersand symbol, `'&'`, in front of the referring shortcut.

The fifth shortcut tells CVS to retrieve the items in project `foobar`. Also, the `-d` option tells the tool to store those items under the directory `Test/foobar`.

Finally, the sixth shortcut tells CVS to retrieve only the file `foobar.css`. And the `-a` option tells the tool to save the file in the directory `foobar/styles`.

Listing 3. Example of a modules file.

```
MyFoo           foobar
MyPics     -a  foobar/graphics
NoPics     -a  !foobar/graphics   foobar/
PicFree         &NoPics
MySite     -d  Test    foobar
MyStyles   -a  foobar/styles/foobar.css
```

Tracking Project Activity

Tracking project activity is an important part of the work cycle. It helps identify which files got the most attention and those that got the least. It helps measure the number of changes made to each file. Tracking also shows which users worked on which files, and the time spent on each file. It can also help users coordinate their efforts and avoid possible conflicts.

Tracking file activity

Use the `cvs log` command to list the changes made to the project or file. For example, to list all the changes to the file `foobar.htm`, type the command as follows.

```
cvs log foobar.htm
```

CVS then displays the list on the Terminal window. To save the list of changes into a file, e.g. `foobar.log`, use the `'>'` operator.

```
cvs log foobar.htm > foobar.log
```

You can restrict the list to a specific date or revision range. To restrict it by date, use the **d** option. For example, to show only changes made between 2007 Aug 1 and 2007 Sep 1, type the command as follows.

```
cvs log -d"2007/08/01<2007/09/01" foobar
```

Notice that there is no space between the -d option and the date range. Also, notice the use of the `'<'` token to specify an exclusive range. For an inclusive date range, retype the command with a `'<='` token.

```
cvs log -d"2007/08/01<=2007/09/01" foobar
```

To restrict the range by revision number, use the `-r` option. For example, to show only the changes made between revisions 1.2 and 1.5, type the command as follows.

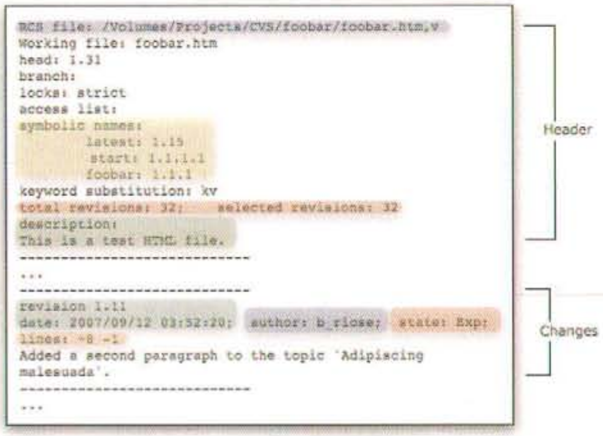
```
cvs log -r1.2:1.5 foobar
```

Notice again that there is no space between the option and the range. Notice also the use of the ':' token to specify an inclusive range.

To learn other useful log options, read the online CVS manual (*see reference*).

Figure 1 is a sample output from the `cvs log` command. The header section tells us where the project file is located on the repository. It tells us what tags belong to the file (*gold*), and the revision count (*red*). It also shows a short description of the file (*green*), if one is available.

Right after the header is a list of changes made to the file. Each entry starts with the revision number and date of the change (*green*). It shows the user who made the change (*blue*), the file's current state (*red*), and the number of lines that changed (*pink*).



```
RCS file: /Volumes/Projects/CVS/foobar/foobar.htm,v
Working file: foobar.htm
head: 1.31
branch:
locks: strict
access list:
symbolic names:
        latest: 1.15
        start: 1.1.1.1
        foobar: 1.1.1
keyword substitution: kv
total revisions: 32;    selected revisions: 32
description:
This is a test HTML file.
----------------------------
...
----------------------------
revision 1.11
date: 2007/09/12 03:52:20;  author: b_riose;  state: Exp;
lines: +8 -1
Added a second paragraph to the topic 'Adipiscing
malesuada'.
----------------------------
...
```

Figure 1. Sample output of the cvs log command.

The output of the `cvs log` command, however, is not customizable. Creating it is a separate step, making it prone to neglect. It can also be changed either by accident or by intent.

But you can have CVS create the log output automatically. You can customize the output to suite your needs. You can also make the output a part of your project file, thus serving as a version history for that file. You can do all these by using *revision keywords*.

Table 1 is a list of common revision keywords. To use these keywords, add them within a comments block. Doing so will tell the file's owner to ignore the keywords. Also, when you commit

Bring your voice to life.
Put the power of speech recognition on your Macintosh with MacSpeech Dictate, and say goodbye to typing.
• Amazing speech recognition accuracy
• Works with the apps you already own
• Requires only minimal training
www.macspeech.com
MacSpeech
Think it. Speak it. Create it.
MacSpeech Dictate
Speech Recognition for Mac OS X
Your Mac is Listening
Macworld BEST OF SHOW 2008

Network, Server and Appliance Monitoring.

For Mac OS X

Xserve RAID

Xserve (Intel & G5)

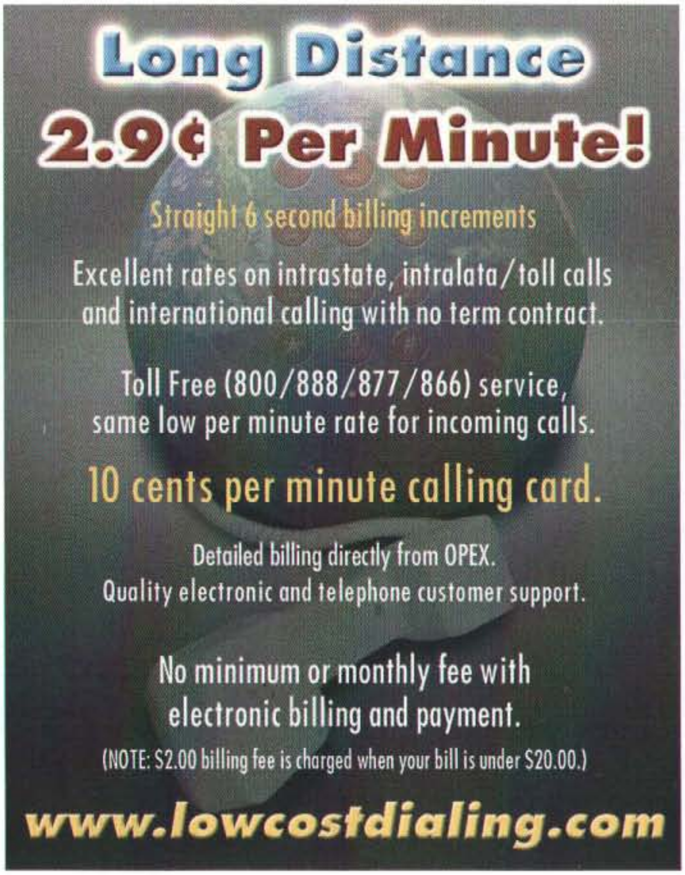
Airport

Lithium Network Monitoring Platform

Lithium can now monitor your Xserve, Xserve RAID, Qlogic switches, Airports, Mac OS X Server... and everything else in your network.

www.lithiumcorp.com

lithium Corp.



the file to the repository, CVS updates each keyword with the right information.

Table 1. A list of useful revision keywords.

Keyword	Description
`$Author$`	Name of the last user who made the change
`$Date$`	Date when the change was made
`$Header$`	Basic header information
`$Log$`	A description of changes made to the file
`$State$`	State of the file when the change was made
`$Source$`	Location of the project file in the repository
`$Revision$`	Revision number assigned to the file

Listing 5 is a sample revision block for an HTML file. Notice the block is placed between the comment marks <!– –>. Assume you added this block to the file **`foobar.htm`**. Then you committed the file by typing the following command.

```
    cvs commit -m "Added the revision history block"
foobar.htm
```

Listing 5. A sample revision block.

```
<!–
    $Source$
    $Author$
    $Date$
    Revision history:
        $Log$
–>
```

CVS responds by first updating the revision block as shown by Listing 6. It then commits the updated file to the repository. Notice that the block retains the keywords, even after the update. Also, future committals will cause CVS to insert more entries right after the `$Log$` keyword.

Listing 6. The revision block after a committal.

```
<!–
    $Source: /Volumes/Projects/CVS/foobar/foobar.htm,v $
    $Author: s_hardin $
    $Date: 2007/09/11 04:30:50 $
    Revision history:
        $Log: foobar.htm,v $
        Revision 1.2  2007/09/11 04:30:50  s_hardin
        Added the revision history block.
–>
```

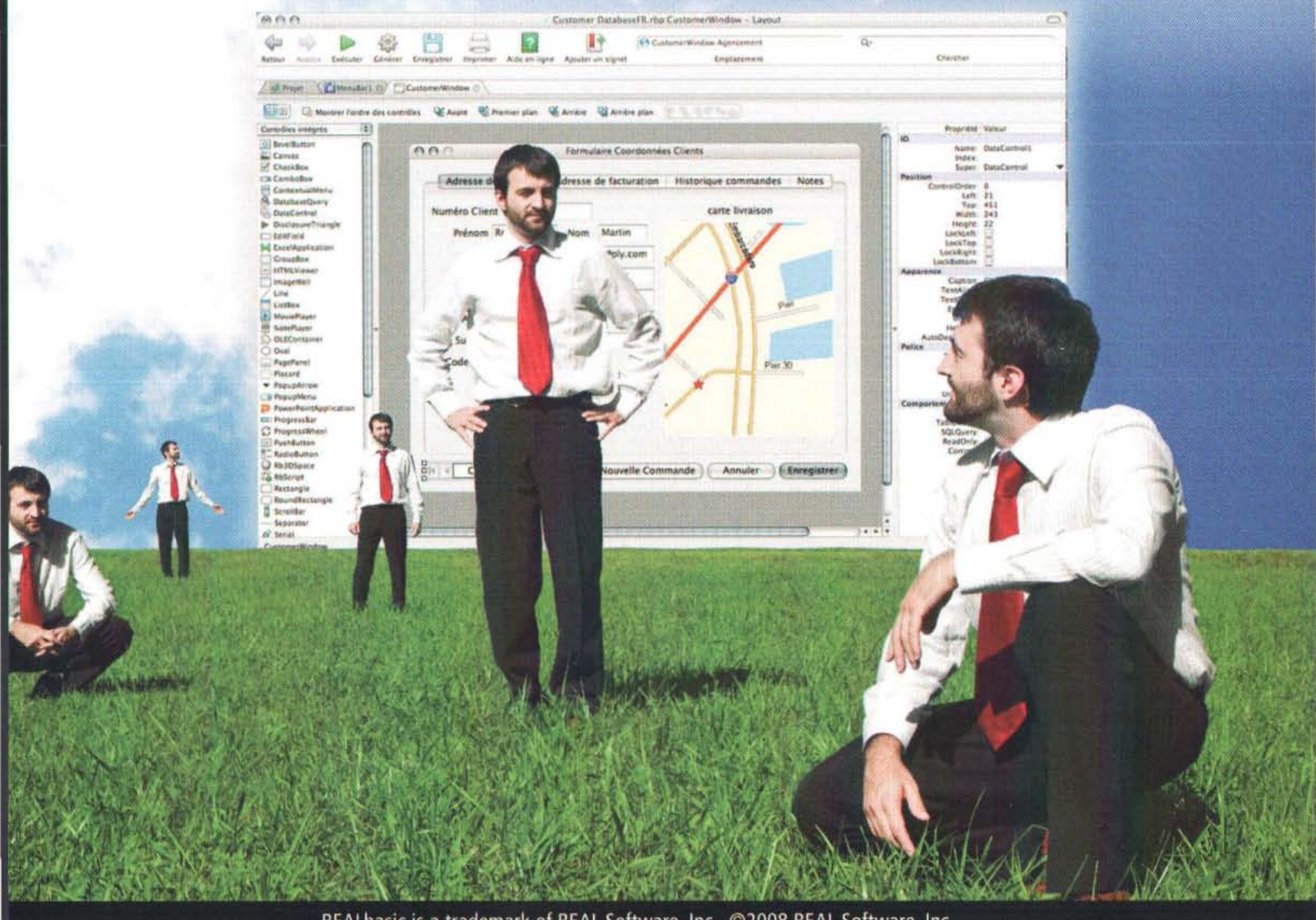
Tracking user activities

Use the **`cvs watch`** command to keep track of users working on the same project. This command performs two functions. First, it sets the project file to a *read-only state* during a checkout or an update. Second, it adds a user to the watch list.

The basic syntax of the **`cvs watch`** command is as follows.

```
    cvs watch operator [file …]
```

The *`operator`* argument sets the behavior of the command. It has four possible values: `on`, `off`, `add`, and `remove`. You can also list more than one file after the command.


Customer DatabaseFR.rbp CustomerWindow - Layout
Formulaire Coordonnées Clients
Adresse de facturation
Historique commandes
Notes
Numéro Client
Prénom
Nom
Martin
carte livraison
Nouvelle Commande
Annuler
Enregistrer
Contrôles intégrés
Propriété
Valeur

Let us assume that you are working on project **foobar** with another user named **h_mallow**. To add the file to the watch list, use the **cvs watch** command with an **on** operator.

```
cvs watch on foobar.htm
```

When h_mallow checks out a copy of the file, he will find the file in a read-only state. To remove the file from the watch list, use the command with an off operator.

```
cvs watch off foobar.htm
```

Since you placed the file on the watch list, you should inform **h_mallow** that you are watching that file. To do so, use the **cvs watch** command with an **add** operator.

```
cvs watch add foobar.htm
```

To remove yourself from the list, use the command with a remove operator.

```
cvs watch remove foobar.htm
```

To get a list of other users watching the file, use the cvs watchers command.

```
cvs watchers foobar.htm
```

Listing 7 shows a sample output of the above command.

Listing 7. Sample output of the cvs watchers command.

```
foobar.css      h_mallow edit     unedit  commit
foobar.htm      youRhere edit     unedit  commit
```

Now assume that user **h_mallow** added the file **foobar.css** to the watch list. When you checked out that file, you will find it set to a read-only state. If you want to change the file, first use the **cvs edit** command to change its state to *read/write*.

```
cvs edit foobar.css
```

When user h_mallow types the cvs watchers command, he gets the following list of users.

```
foobar.css        h_mallow edit     unedit  commit
      youRhere tedit   tunedit tcommit
foobar.htm        youRhere edit     unedit  commit
```

He then sees that you (**bold**) have started work on the foobar.css file.

Once you have made your changes to **foobar.css**, use the **cvs unedit** command to set the file's state back to read-only.

```
cvs unedit foobar.css
```

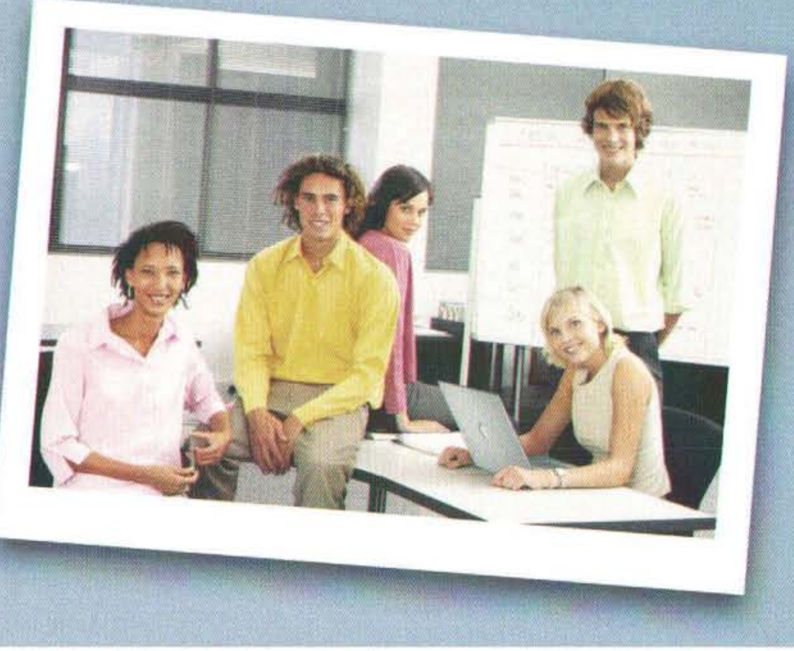
Committing your changes to the repository will also do the same thing. And both actions will remove you from the watch list.

Managing The Repository

Sometimes, you want to make minor changes to your project repository. You may want to set tags to specific project revisions. Or you may want to generate a patch for a specific project file.

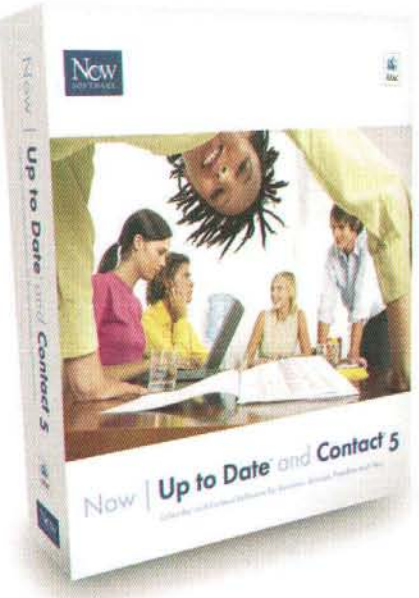
CVS gives you the means to make these changes directly on the repository. But be aware that you could easily mess up the repository as well. Make sure to create a backup of your repository beforehand. Also, make sure to allow only a small group of trusted users to change the repository itself.

Now **scheduling** and **contact management** for your entire organization.



Now | **Up to Date**® and **Contact**® 5

Calendar and Contact Software for Business, Groups, Families and You.



Is this project on schedule? When are you available to meet about the systems upgrade? Where are all the field techs today? When was the last time anyone talked to our biggest customer?

Virtually all groups live (or die) by their abilities to meet deadlines and keep track of their customers, prospects, and vendors. Few small companies or even departments of big companies have the tools they need.

Now Up-to-Date & Contact might just be the calendar and contact software for you. It's time-tested and used by more Mac-based companies than any other solution. And it's cross-platform—available for your PC users, too. It's easy to install and manage and simple for your employees to understand and use.

Using Now Up-to-Date & Contact you can schedule meetings for multiple users, view multiple, simultaneous calendars, and reserve rooms and resources. You can share contact information about your customers, prospects and vendors. And using our free server software you can set it up in minutes and share with users in the office or from anywhere with an internet connection.

Call us now at 866-527-0556 or email us at mactech@nowsoftware.com and we'll send you our free evaluation kit, including the book that will make it all easy, "Take Control of Now Up-to-Date & Contact" from Take Control books!

Now SOFTWARE

Phone: 866-527-0556

Web: www.nowsoftware.com

Managing tags

Now, you use the `cvs tag` command to assign a tag to a file or project from your working copy. To do the same directly on the repository, use the `cvs rtag` command. Unlike the former, you do not need a working copy of the project to use this command.

The basic syntax of the `cvs rtag` command is as follows.

```
cvs rtag [options] tag file | directory | project
```

This command shares many of the same options as the `cvs tag` command. For example, that you are managing the project `foobar`. To set the tag `alpha1` to the *latest revision* of the file `foobar.htm`, type the command as follows.

```
cvs rtag alpha1 foobar.htm
```

To set the same tag to the latest revision of project `foobar`, type it as follows.

```
cvs rtag alpha1 foobar
```

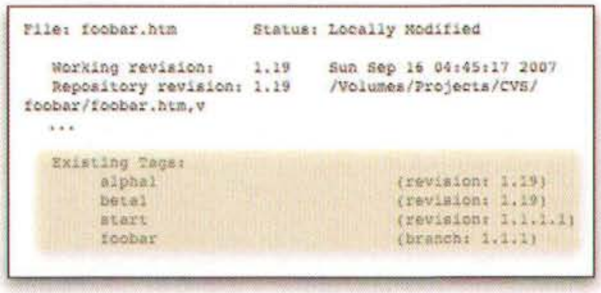
Suppose a user gave the wrong tag `beta1` to the latest revision of `foobar.htm`. To remove that tag, use the `-d` option.

```
cvs rtag -d beta1 foobar.htm
```

But make sure that the tag exists before you remove it. Otherwise, CVS will return an error message. Use the `cvs status` command with a `-v` option to get a list of tags.

```
cvs status -v foobar.htm
```

CVS then displays a list of all tags assigned to the file `foobar.htm` (Figure 2, *gold*).



```
File: foobar.htm            Status: Locally Modified

   Working revision:    1.19     Sun Sep 16 04:45:17 2007
   Repository revision: 1.19     /Volumes/Projects/CVS/
foobar/foobar.htm,v
   ...

   Existing Tags:
        alpha1                              (revision: 1.19)
        beta1                               (revision: 1.19)
        start                               (revision: 1.1.1.1)
        foobar                              (branch: 1.1.1)
```

Figure 2. Sample output of the cvs status -v command.

Assume you want to set the tag `dev1` to revision 1.15 of `foobar.htm`. To do so, use the `cvs rtag` command with a `-r` option.

```
cvs rtag -r 1.15 dev1 foobar.htm
```

If you want to set the same tag to the file `foobar.css` revised on 2007 Sept 01, use the `-D` option.

```
cvs rtag -D 2007/09/01 dev1 foobar.css
```

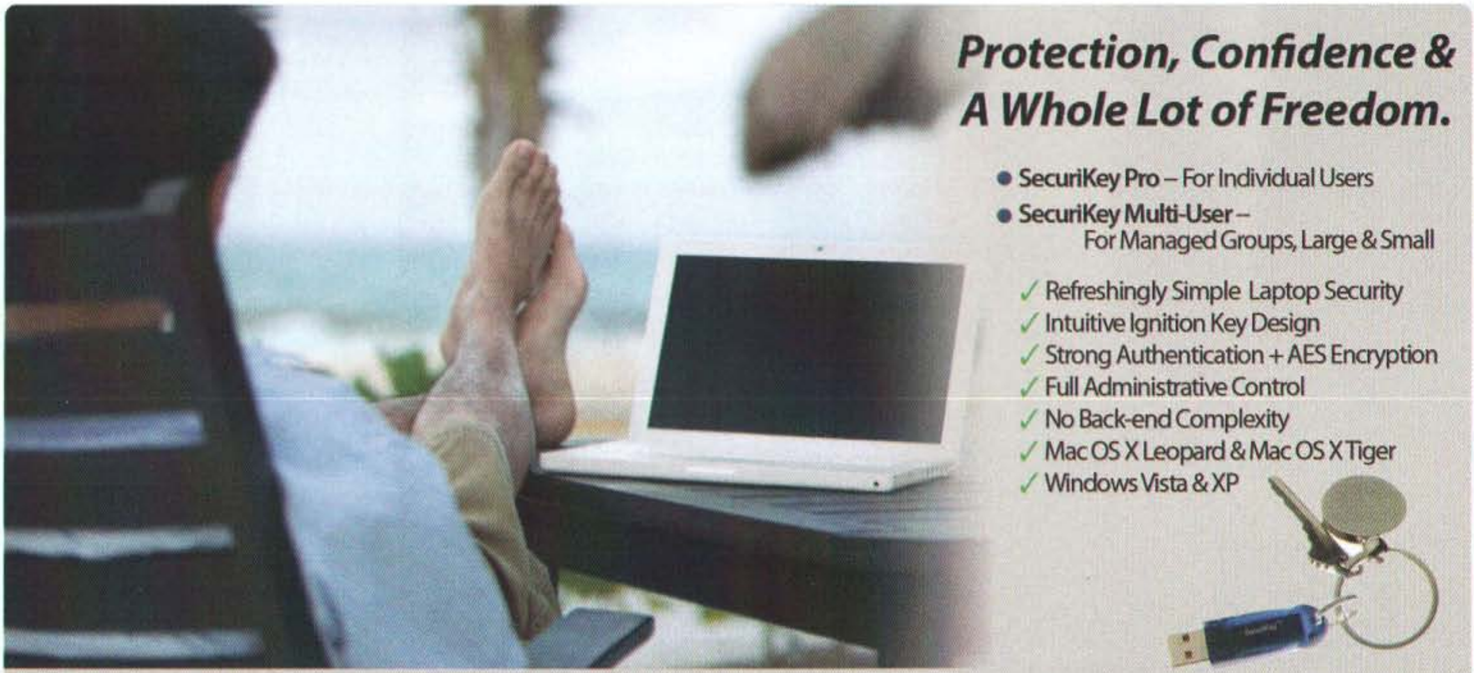
Again, make sure that either revision number or date exists on the repository. If you are not sure, add a `f` option to the command.

```
cvs rtag -f -r 1.15 dev1 foobar.htm
```

If CVS fails to find the specified revision or date, it will set the tag to the latest revision of that file.

Suppose a user sets the tag `GM1` to the wrong revision of `foobar.htm`. To move that tag to the correct revision, e.g. 2.0, use the `-F` option.

```
cvs rtag -F -r 2.0 GM1 foobar.htm
```



First, CVS removes the tag GM1 from foobar.htm. Then it assigns the same tag to revision 2.0 of that file.

Assume that project foobar has a subdirectory named **scripts**. To set the tag **alpha3** to all the files in that subdirectory, type the **cvs rtag** command as follows.

```
cvs rtag alpha3 foobar/scripts
```

Not only will CVS tag all the files in scripts, it will also locate any scripts subdirectories and tag their files as well. To tag only those files in the scripts subdirectory, use the -l option.

```
cvs rtag -l alpha3 foobar/scripts
```

Managing diffs

Now, you use the **cvs diff** command to compare two revisions of a file or project. But to do the comparisons directly on the repository, use the **cvs rdiff** command. Again, unlike the former, you do not need a working copy of the project to use this command.

The basic syntax of the **cvs rdiff** command is as follows.

```
cvs rdiff [options] file | project
```

This command shares many of the same options as the cvs rdiff command. Assume, for example, that you are managing the project foobar. To set the tag alpha1 to the latest revision of foobar.htm, type the command as follows.

Assume you are working on project **foobar**. To compare its latest revision against revision 1.5, type the **cvs rdiff** command with a **-r** option.

```
cvs rdiff -r 1.5 foobar
```

To compare revision 1.1 against 1.5, use two -r options as follows.

```
cvs rdiff -r 1.1 -r 1.5 foobar
```

Make sure the repository contains both revisions, or CVS will return an error message. If you are not sure, you can add a -f option to the command.

```
cvs rdiff -f -r 1.1 -r 1.5 foobar
```

CVS will use the latest revision if it fails to find one of the revision numbers.

Suppose you want to check on the project file **foobar.htm**. To compare its latest revision against the one on 2007 Sep 01, type the **cvs rdiff** command with a -D option.

```
cvs rdiff -D 2007/09/01 foobar/
```

To compare the file from 2007 Sep 15 against the one on 2007 Sep 01, use two -D options as follows.

```
cvs rdiff -D 2007/09/01 -D 2007/09/30 foobar/foobar.htm
```

Make sure to place the earliest date first. Make sure also that the repository contains both dates. Use the f option if you are not sure.

Now the **cvs rdiff** command generates its output in one of two formats. The first format, *context diff*, is the default format. It is the one used by most BSD projects. The second format, *unified diff*, is a more compact format. It is also the one used by most GNU projects.

Assume you are comparing the latest revision of **foobar.css** against revision 1.5. To save the results in context diff, type the **cvs rdiff** command as follows.

```
cvs rdiff -r 1.5 foobar/foobar.css > foobar.diff
```

CVS then saves the comparison results into the file foobar.diff. To save the results in unified diff, add a -u option.

```
cvs rdiff -r 1.5 -u foobar/foobar.css > foobar.diff
```

Once you saved the results, you can send the file to non-CVS users. Those users can then update their copy of `foobar.css` using your diff file and the *patch* tool. For more information about that tool, type `man patch` at the Terminal prompt.

Managing miscellanea

Use the **`cvs admin`** command to make changes to a project file in your repository. Unlike the previous two commands, you will need a working copy of the project in order to use this command. Also, most changes you make with this command are *undoable. Make a backup of the repository to protect yourself from any mistakes.*

The basic syntax of the **`cvs admin`** command is as follows.

```
cvs admin [options] [file]
```

The file argument specifies which file to change. If this argument is missing, the command will apply the change to all the *latest revisions* of the project's files. The *`options`* argument specifies what changes to make to the file in the repository.

Suppose a user used the wrong message when he committed his changes to the file **`foobar.htm`**. To correct the message, first find out the revision number of that committed file, e.g. 1.5. Then use the **`cvs admin`** command with a **`-m`** option as follows.

```
    cvs admin -m1.5:"the correct committal message here"
foobar.htm
```

Suppose you want to trim down your project repository. A good way to do so is to use the **`cvs admin`** command with a `-o` option. Then specify a range of revisions to remove from the repository. For example, to remove all revisions of **`foobar.htm`** *between 1.2 and 1.5*, type the command as follows.

```
cvs admin -o1.2::1.5 foobar.htm
```

Note the use of double-colons, '`::`', when setting the range. To remove revisions *1.2 and 1.5 as well*, use a *single colon*, '`:`', to set the range.

```
cvs admin -o1.2:1.5 foobar.htm
```

To remove revisions *1.2 and older* of `foobar.htm`, specify the range as follows.

```
cvs admin -o:1.2 foobar.htm
```

To remove revisions *1.2 and newer* of that file, specify the range as follows.

```
cvs admin -o1.2: foobar.htm
```

Again, if you want to exclude revision 1.2 from the range, use *double-colons*, '`::`', to set the range.

By default, CVS sets all project files to an `Exp` (*experimental*) state. Other possible states include `Stab`, for *stable*; and `Rel`, for *release*. You can also specify your own state label.

Suppose you want to change the latest revision of the **`foobar.htm`** file to a **`Stab`** state. To do so, use the **`cvs admin`** command with a **`-s`** option.

```
cvs admin -sStab: foobar.htm
```

To set revision 1.5 of `foobar.htm` to a `Test` state, type the command as follows.

```
cvs admin -sTest:1.5 foobar.htm
```

Notice that there are no spaces between the `-s` option and the state label itself. Notice also that the revision number comes after the '`:`' token. To find out if CVS set the correct state, type `cvs log foobar.htm`. CVS will display the file's state in one of its log entries (Figure 1, *red*).

CVS also leaves the project file's description *blank*. To add your own description, use the **`cvs admin`** command with a **`-t`** option. For example, to add a description to **`foobar.htm`**, type the command as follows.

```
cvs admin -t-"This is a test HTML file." foobar.htm
```

Note the use of a dash, '`-`', to separate the `-t` option and the description string. To add a much larger description, first add the description to a separate file, e.g. `foobar.txt`. Then type the `cvs admin` command as follows.

```
cvs admin -tfoobar.txt foobar.htm
```

Also, when you type `cvs log foobar.htm`, you will see the description string on that command's output (Figure 1, *green*).

Final Thoughts

This article has shown you a handful of ways to get more out of your CVS setup. It showed how to configure your setup to suit your needs. It showed how to track changes to project files and the users that made them. It also showed how to maintain the repository, its tags and diffs.

Despite its limits, CVS is still a popular tool for managing OS X projects. It is available on all versions of OS X, and it integrates well with other tools such as BBEdit and Xcode. It also inspired users to create tools that address its limits and extend its capabilities.

It is possible that other SCM tools such as Subversion and Git will finally replace CVS in the near future. Until that happens, however, CVS is a strong management system, alive and well.

Bibliography and References

Cruz, José R.C. "Version Control on a Budget". *MacTech Magazine*. Volume 22. Issue 10. (c) 2006 MacTech.

Cederqvist, Per. "Reference manual for Administrative files". *CVS—Concurrent Versions System v1.11.22*. (c) 2003, 2006 Ximbiot, LLC. Online:
http://ximbiot.com/cvs/manual/cvs1.11.22/cvs_18.html#SEC158

Fogel, Karl, Moshe Bar. "Using Keyword Expansion". *Open Source Development with CVS, 3rd Edition*. (c) 1999, 2000 Karl Fogel. Online:
http://cvsbook.redbean.com/cvsbook.html#Using%20Keyword%20-Expansion

Free Software Foundation. "Pattern Matching". *Bash Reference Manual*. (c) 1991-2002 Free Software Foundation. Online:
http://www.gnu.org/software/bash/manual/bashref.html#SEC35

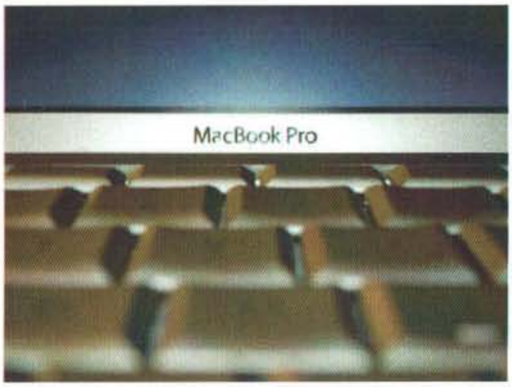
Wikipedia. *Diff*. (2007 Oct 6). Copyright 2007. Wikipedia Foundation, Inc. Online: http://en.wikipedia.org/wiki/Diff.

About The Author

JC is a freelance engineering writer who lives happily in North Vancouver, British Columbia. He divides his time between writing technical articles, and teaching origami at his district's public library. He can be reached at anarakisware@gmail.com.

Short Term Computer Needs?

- Latest Apple Technology
- Legacy Macs
- Windows Computers



- Rent One or Hundreds
- Audiovisual Gear
- Projectors, Plasmas, LCDs
- B&W & Color Copiers

m

Mac Rentals
Event Technology Services

- Name Brands
- Nationwide Delivery
- Professional Setup
- Onsite Technicians
- Same Day Service

- Compatibility & Compliance Testing
- Short Term Projects, Production Needs
- Trade Shows, Conventions, Corporate Events
- Trainings Sessions, Classrooms
- Fortune 100 and Educational P.O.'s Accepted!

Toll Free
800-756-6227
Outside US/Canada: 858-454-8535

For Online Quote Request, Visit:

www.madcomputing.com

Largest Apple Rental Inventory in the U.S.!

Book Review

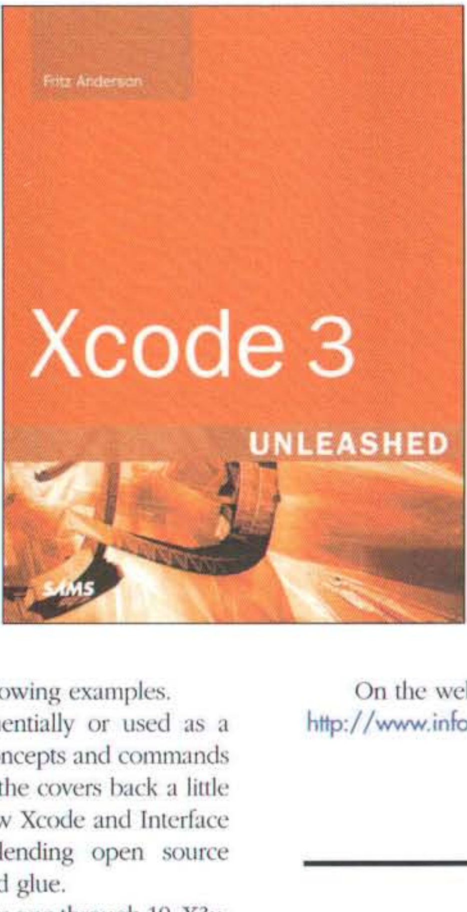
by Edward Marczak

Xcode 3 Unleashed, by Fritz Anderson

Rev up your Xcode-fu

Xcode 3 Unleashed fills a unique niche in the Macintosh development world. It doesn't aim to teach coding per se, but you'll probably pick up some new techniques by reading the book. Nor does it aim to teach Cocoa or Objective-C, but you'll see plenty. Xcode 3 Unleashed (X3u) is just that: an exposé on Xcode, Apple's free development environment for OS X. As a freely available system, Xcode may be a touch under-documented, or, perhaps, not documented in a manner conductive to learning about each of its many facets. X3u solves that.

X3u is divided into 2 main parts, with part 3 containing appendices. The first part is meant to be read sequentially, and leads the reader through writing a sample application using Xcode. The application starts as a simple command-line tool, and by the end of the section is an advanced GUI application that has a command-line helper and relies on a custom framework library. Each step demonstrates a clear how-to using the Xcode IDE. My favorite part of the first section is how early Fritz introduces version control–using subversion–gives every reason why it's critical to use version control, and then integrates the use of subversion into Xcode and the following examples.



The second part can be read sequentially or used as a reference. This section gets deeper into concepts and commands presented into the first section, and pulls the covers back a little further. Here, Fritz presents details on how Xcode and Interface Builder work most of their magic, blending open source components and Apple supplied tools and glue.

In part 1, which encompasses chapters one through 19, X3u: shows how to launch Xcode and write and build a "Hello, World!" application (chapter 1), shows how to use Xcode's integrated debugger (chapter 3), explains what happens during a compile and build (chapter 4), introduces version control (chapter 8), details property lists (chapter 9), creates a library for the sample project (chapter 10), explains how to write unit tests for your application using the SenTestingKit framework (chapter 12), details documentation–both Xcode-wise and for purposes of documenting your source (chapter 15), explains how to target different architectures (chapter 17) and more.

As mentioned, part 2–chapters 20 through 27–refine the lessons taught in part 1 and also go deeper into Xcode and Interface Builder. Chapter 20 opens with details about Xcode that will make your job easier, and knowledge greater. It touches on Code Sense and indexing, Code Focus, gives a closer look at the Groups and Files list in the editor window and more. From there, chapter 21, "Xcode for make Veterans," and chapter 24, "A Legacy Project," takes an open source application and shows how to build an Xcode project from the download's Makefile and other files (practically worth the price of the book alone). Part 2 continues on with: more on debugging (chapter 22) and more ways to introspect and improve an application using Apple supplied tools such as Shark and CHUD (chapter 25) and Instruments (chapter 26).

Simply, Xcode 3 Unleashed is for anyone that uses Xcode. The reader learns the rhythm of the development cycle of an OS X application developed with Xcode. Fritz's knowledge and experience is clear. The writing, presentation and editing are all excellent. Each chapter is tightly focused and concise. X3u targets Xcode 3, but does *not* solely target OS X Leopard, v10.5. Fritz does, however, always point out where there would be a difference in doing so. In short, there's something for everyone, no matter their level, beginner or long-time Xcode user, and belongs in every OS X or iPhone developer's library.

Title: *Xcode 3 Unleashed*
Author: Fritz Anderson
Publisher: Sams
Price: $44.99 USA
ISBN-13: 978-0-321-55263-1

On the web:
http://www.informit.com/title/9780321552631

About The Author

Ed Marczak is the Executive Editor for MacTech Magazine, and has been lucky enough to have ridden the computing and technology wave from early on. From teletype computing to MVS to Netware to modern OS X, his interest was piqued. He has also been fortunate enough to come into contact with some of the best minds in the business. Ed spends his non-compute time with his wife and two daughters.

Advertiser/Product Index

The index on this page is provided as a service to our readers.
The publisher does not assume any liability for errors or omissions.

THE MACTECH SPOTLIGHT

Blair Yakimovich

TRANSGAMING, INC.

What do you do?

I'm an Associate Producer for TransGaming Inc., an innovative company who provides portability solutions for video game publishers and developers.

How long have you been doing what you do?

I've been producing for a year. Before that I was a developer for 7 years.

What was your first computer?

In 1990, my parents bought a Mac classic for the family. My life was then devoted to acquiring as many shareware games as I could get my eleven-year old paws on. Nightly, I thanked Silicon Beach Software for Dark Castle and Beyond Dark Castle.

Are you Mac-only, or a multi-platform person?

I'm a multi-platform person, but I predominantly use the Mac. I was raised on the Mac for eight years before I was introduced to Windows in high school. My education in programming taught me to appreciate Linux. Today, I find I can enjoy the best of many platforms through the Mac - the classic user-environment of a Mac, the power of the command line in the Terminal, and a rapidly growing host of games seen previously only on Windows.

What's the coolest thing about the Mac?

I was going to say the chime that sounds every time you turn one on - it definitely evokes a Pavlovian response of excitement from me. My Windows PC beeps...it doesn't chime. Expose is pretty rad too.

What is the advice you'd give to someone trying to get into this line of work today?

Enthusiasm - let it fuel your desire to break in. If you're a developer, develop small games and demos to show off your chops. If you love to play games and have an analytical mind, QA is a great place to test your mettle.

What's the coolest tech thing you've done using OS X?

Make PC games work on the Mac.

Where can we see a sample of your work?

At http://www.gametreeonline.com and the Apple Store! I've worked on over 20 games for the Mac, including Spore, Jade Empire, Battlefield 2142, Command and Conquer 3: Tiberium Wars, Need for Speed Carbon, and The Lost Cases of Sherlock Holmes.

The next way I'm going to impact IT/OS X/the Mac universe is:

To continue to take my passion for Mac gaming and infect interested publishers and developers. The Mac Gaming community is rapidly expanding, and I want to continue to be at the forefront, enabling as many titles as I can using our Cider technology. My goal is for no Mac-user to walk into an electronics store eager for games, and leave empty-handed.

If you or someone you know belongs in the MacTech Spotlight, let us know! Send details to editorial@mactech.com

Now Featuring

The perfect gifts for the holidays *(or any day).*

» Five gifts to give this year.

#1 17" MacBook Pro
#5 Final Cut Studio
Final Cut Studio
#2
#4
AppleCare Protection Plan
AppleCare Protection Plan
Music
#3 iPod nano (one in every color?)
EZQuest Thunder Pro 500GB Quad Interface HD

With over 3,000 featured products year-round, Small Dog Electronics has a gift for everyone. From **iPods** to **pro video cards** to **Mac bundles** that feature everything you could possibly want in one package, **Smalldog.com** is the Mac lovers' destination for hardware, software and more.

Buying for yourself? Either way, visit **Smalldog.com/specials** for the latest deals–browse from over 75 featured specials a day! Every order outside of Vermont is always **tax-free,** and our friendly, knowledgeable staff is always ready to help with any question you may have.

Small Dog Electronics
Always By Your Side

- over 3,000 featured products
- 5-star online merchant rating
- tax-free outside of VT

www.smalldog.com 800-511-MACS Apple Specialist